

# Módulo 10

## Sistemas de Gestión Empresarial

```

260 function updatePhotoDescription() {
261     if (descriptions.length > (page * 9) + (currentImage.substring(0, 1))) {
262         document.getElementById("bigImageDesc").innerHTML = descriptions[page * 9 + currentImage.substring(0, 1)];
263     }
264 }
265
266 function updateAllImages() {
267     var i = 1;
268     while (i < 10) {
269         var elementId = 'foto' + i;
270         var elementIdBig = 'bigImage' + i;
271         if (page * 9 + i - 1 < photos.length) {
272             document.getElementById(elementId).src = 'images/mini/' + photos[page * 9 + i - 1];
273             document.getElementById(elementIdBig).src = 'images/webcam/' + photos[page * 9 + i - 1];
274         } else {
275             document.getElementById(elementId).src = 'images/mini/';
276             document.getElementById(elementIdBig).src = 'images/webcam/';
277         }
278         i++;
279     }
280 }

```

## UF1. SISTEMAS ERP- CRM. IMPLANTACIÓN.....3

<b>1.</b>	<b>Identificación de sistemas ERP- CRM. ....</b>	<b>4</b>
1.1.	Concepto ERP (Planificación de los recursos empresariales). ....	6
1.2.	Revisión de los ERP actuales.....	8
1.3.	Concepto de CRM (sistemas de gestión de relaciones con clientes). ....	14
1.4.	Revisión de los CRM actuales. ....	15
1.5.	Concepto de data warehouse (Almacenes de datos). ....	16
1.6.	Revisión de los data warehouse actuales y posiblemente incorporados a los sistemas ERP- CRM. ....	16
1.7.	Sistemas gestores de bases de datos compatibles con el software. ....	19
1.8.	Verificación de la instalación y configuración de los sistemas operativos y de gestión de datos. ....	21
<b>2.</b>	<b>Instalación y configuración de sistemas ERP- CRM .....</b>	<b>24</b>
2.1.	Tipo de licencia. ....	24
2.2.	Tipo de instalación. Monopuesto. Cliente/ Servidor.....	25
2.3.	Procesos de instalación del sistema ERP- CRM. ....	27
2.4.	Módulos, Gestión y Actualización de un sistema ERP- CRM. ....	33
2.5.	Administración básica y configuración .....	40

## UF2. SISTEMAS ERP- CRM. EXPLOTACIÓN Y ADECUACIÓN. ....46

<b>1.</b>	<b>Organización y consulta de la información. ....</b>	<b>46</b>
1.1.	Introducción a las bases de datos. ....	46
1.2.	Interfaces de entrada de datos y de procesos. ....	51
1.3.	Cálculos de pedidos, albaranes, facturas, asientos predefinidos, trazabilidad, producción. ....	60
1.4.	Gestión de los recursos humanos ....	67
1.5.	Herramienta de Monitorización. Incidencias: resolución e identificación. ....	72
1.6.	Procesos de extracción de datos en sistemas ERP- CRM y almacenes de datos. ....	77
<b>2.</b>	<b>Implantación de sistemas ERP. CRM en una empresa.....</b>	<b>81</b>
2.1.	Metodología de implantación .....	83
2.2.	Tipo de empresa. Necesidades de la empresa. ....	88
2.3.	Selección de los módulos del sistema ERP- CRM. ....	88
2.4.	Tablas y vistas a adaptar.....	92
2.5.	Consultas necesarias para obtener información. ....	97
2.6.	Creación de formularios personalizados. ....	97
2.7.	Creación de informes personalizados. ....	97
2.8.	Creación de cuadros de mando personalizados.....	101
<b>3.</b>	<b>Desarrollo de componentes.....</b>	<b>102</b>
3.1.	Lenguaje proporcionado por los sistemas ERP- CRM. Características y sintaxis del lenguaje. Declaración de datos. Estructuras de programación. Sentencias del lenguaje. ....	103
3.2.	Entornos de desarrollo y herramientas de desarrollo en sistemas ERP y CRM.....	119
3.3.	Operaciones con Datos en los objetos .....	121
3.4.	Extracciones de informaciones contenidas en sistemas ERP- CRM, procesamiento de datos. ....	122
3.5.	Llamadas a funciones, librerías de funciones (API).227 .....	122
3.6.	Depuración de un programa. ....	123
3.7.	Manejo de errores.....	124

## BIBLIOGRAFÍA ..... 127

## UF1. Sistemas ERP- CRM. Implantación.

Los sistemas de gestión empresarial abarcan una gran cantidad de procesos y diferentes técnicas que se han ido desarrollando con el paso de los años.

En esta unidad formativa, nos vamos a centrar principalmente en la introducción a la gestión. Además, vamos a enumerar y repasar aquellos conceptos básicos que debemos tener en cuenta en una empresa. Son los conocimientos mínimos que debemos tener para comprender el funcionamiento de la misma.

Cuando hablamos de la gestión empresarial, no pretendemos obtener valores exactos sobre algo determinado, sino poder obtener una **visión global de una empresa, junto con sus componentes principales y procesos**, para conseguir entender de forma correcta el funcionamiento de la empresa como un conjunto y saber tomar las decisiones más acertadas en cada momento, sobre todo, cuando tengamos que decidir qué sistema de gestión empresarial debemos implantar.



# 1. Identificación de sistemas ERP- CRM.

Entendemos gestión empresarial como la actividad empresarial que, mediante un conjunto de individuos especializados, como pueden ser: gerentes, consultores, productores, etc. pretende conseguir mejorar la productividad y la competitividad de una determinada empresa.

Entre sus principales características, podemos indicar algunas de las más importantes:

- Una empresa existirá siempre que obtenga beneficio.
- La empresa debe realizar una buena gestión de sus recursos disponibles para ser competitiva.
- El objetivo principal de una empresa debe ser siempre el cliente.

## La informática en la gestión empresarial

Podemos definir que una de las principales funciones de la informática está centrada en el tratamiento automático de la información.

Si las empresas van a desarrollar actividades como pueden ser la generación, manejo y análisis de una determinada información, es lógico que pensemos que la informática va a desarrollar un papel bastante importante en lo que a la gestión empresarial se refiere.

## Concepto de ERP- CRM

- **ERP** (*Enterprise Resource Planning*) Sistema de planificación de recursos empresariales



Este sistema se basa en la planificación, modelado y automatización de un gran número de procesos de una determinada empresa, como pueden ser el área de finanzas, comercial, logística, producción, contabilidad, etc.

Su función principal es unificar y ordenar toda la información de la empresa en un mismo lugar, ayudando de esta forma a facilitar las diferentes tareas.

Destacar también que ERP utiliza una arquitectura modular y, cada uno de los módulos, se encarga de gestionar un área determinada: comercial, producción, logística, finanzas, stock, etc.



Módulos del sistema ERP

- **CRM** (*Customer Relationship Management*) Gestión de las Relaciones con el Cliente

CRM hacer referencia a la estrategia de negocios que se centra específicamente en el cliente.



- **Características**



*Características de los sistemas ERP*

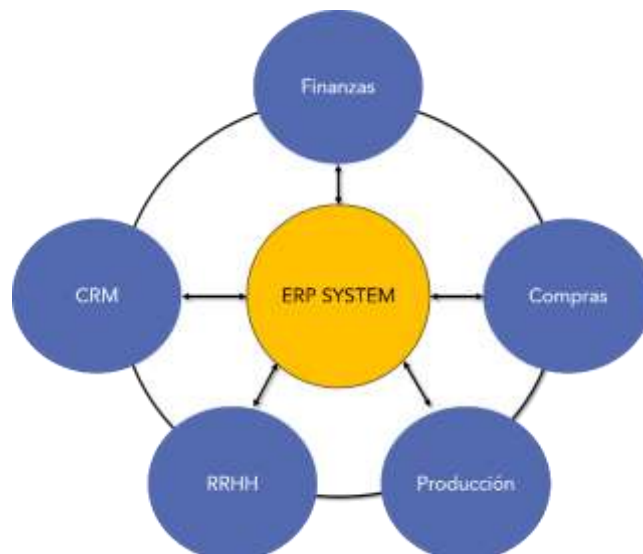
Entre sus principales características, destacamos las siguientes:

- **Integración.** Los sistemas ERP contienen la mayoría de las áreas de las empresas: comercial, logística, producción, contabilidad, etc.
- **Modularidad.** Cada módulo del sistema ERP permite gestionar una determinada área de la empresa: finanzas, logística, producción, etc.
- **Adaptabilidad.** Mediante la unión de las dos características anteriores, podemos lograr una adaptabilidad a las distintas necesidades de una determinada empresa.

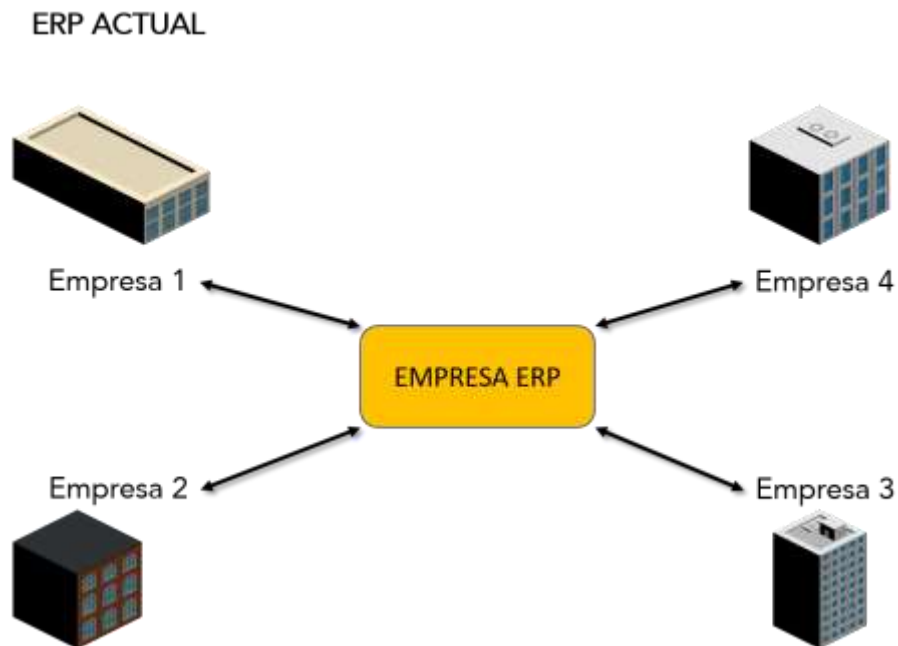
### 1.1. Concepto ERP (Planificación de los recursos empresariales).

Aunque su definición es bastante amplia, nos vamos a centrar en la incorporación de un determinado sistema informático en todos los procesos (internos y externos) que estén relacionados con la empresa.

Estos sistemas permiten ofrecer ayuda a las distintas empresas de forma general y a la toma de decisiones en particular.



A lo largo de los años, los ERP han ido evolucionando hasta conseguir ofrecer a las distintas empresas una serie de beneficios, como que, toda la información de los procesos, se organiza en tablas estáticas o dinámicas utilizando una serie de herramientas que le van a permitir relacionar la información de manera eficiente ante un proceso de análisis y decisión.



**Hoy en día, el ERP engloba tanto los procesos externos como internos de la gestión, abarcando las diferentes necesidades de la pequeña (PYME) y gran empresa en productos ya unificados.**

Estos ERP necesitan:

- Poder gestionar la gestión de procesos de comercio electrónico
- La gestión de los diferentes clientes (CRM)
- Gestión de la cadena de suministro (SCM)
- Gestión de relaciones con proveedores (SRM)
- Inteligencia de negocio (BI)
- Base de conocimiento (KM)
- Gestión de relaciones con socios (PRM)
- Ciclo de vida de un producto (PLM)



Mediante la utilización de todos ellos, vamos a conseguir hacer más fácil el flujo de información entre las empresas relacionadas, aumentando de esta forma la colaboración entre todas ellas.

## 1.2. Revisión de los ERP actuales.

Existen un gran número de soluciones ERP en el mercado, de tal forma que sería imposible hacer referencia a todas. Para llevar a cabo esta unidad formativa, nos basaremos en aquellas soluciones más comunes que podemos encontrar en todo el mundo.

- **SAP**



Creado en Alemania alrededor de los años 70 para desarrollar distintas soluciones empresariales a nivel europeo, extendiéndose con el paso de los años por todo el mundo, convirtiéndose en líder por ventas de soluciones ERP.

Ofrece diferentes soluciones dependiendo del tipo de empresa y de su tamaño, a través de varios productos:

1. **SAP Business Suite.** Este producto fue diseñado para medianas y grandes empresas incorporando una serie de productos básicos y específicos de un determinado sector de una empresa.

Ofrece la posibilidad de interconectar con otro software SAP o de proveedores diferentes. Creado principalmente para favorecer los diferentes procesos de finanzas, fabricación, ventas y gestión de la cadena de suministros y recursos humanos, entre otros.

2. **SAP Business One.** Producto diseñado para pequeñas empresas y añade todos los elementos que se van a necesitar para la gestión,



como pueden ser: ventas, clientes, finanzas, etc. Es bastante rápido a la hora de implantarlo.

Además, esta aplicación, también cuenta con diferentes servicios como pueden ser gestionar la contabilidad y finanzas, gestión de relaciones con el cliente, gestión de compras y operaciones y gestión de informes.

3. **SAP Business All- in- One.** Se presenta como una solución muy completa para la empresa, añadiendo todos los aspectos que se requieren hoy en día. Cuenta con una arquitectura modular en la que el cliente puede ir adaptando sus necesidades. Incorpora como base ERP, CRM, BI, distintas funcionalidades para un determinado sector empresarial y tecnología SAP NetWeaver.
4. **SAP Business ByDesign.** Este software se utiliza, sobre todo, para gestiones empresariales que estén basadas en aplicaciones online. Además, incorpora contabilidad y finanzas, recursos humanos, CRM, ERP entre otros.

- **Oracle**



Creada a finales de los años 70 para llevar a cabo los distintos productos referentes de las bases de datos. En las últimas décadas, se ha ido convirtiendo en un líder indiscutible en su sector.

Sobre el 2005, fue modificando su estrategia añadiendo nuevas empresas que estuvieran relacionadas con los sistemas empresariales que fueran competidores con SAP. Desde ese momento, son las dos empresas que más facturan en el entorno empresarial ERP.

El producto integral ofrecido por Oracle es *JD Edwards Enterprise One*, que cuenta con toda la lógica necesaria para llevar a cabo la gestión integral de

las empresas. Además, permite ofrecer soluciones individuales de ERP, CRM y BI entre otros.

- **Microsoft**



Es una de las empresas de software más conocidas, que en 2001 creó una nueva línea de negocio más orientada al sistema de gestión empresarial.

Este producto se ha ido desarrollando con el paso de los años para ofrecer soporte a medianas empresas (Dynamics NAV) y poder añadir distintas funcionalidades de un ERP actual.

- **OpenBravo**



Creado sobre los años 90 se centra en conseguir orientar su interfaz a navegadores web en vez de a clientes gráficos.

OpenBravo basa su proyecto en dos tipos de proyecto diferentes:

1. Uno que desarrolla la comunidad de licencia libre (OpenBravo Public License).
2. Y otro propietario.

Es el único ERP que tiene su origen en España y ha tenido una gran implantación en todo el mundo.

Entre sus módulos principales podemos encontrar los de ventas, compras, proyectos, CRM, etc. Además, existe una gran cantidad de módulos que incorporan versiones de pago.

OpenBravo cuenta con una arquitectura cliente que permite la integración con otros productos OpenSource existentes.

Por último, también debemos comentar que OpenBravo está capacitado para distribuir un software que sea capaz de realizar la gestión de un punto de venta para cualquier empresa hotelera o comercial por medio de un software denominado OpenBravo POS que se integra con el ERP.

- OpenERP



Este proyecto nace como OpenSource y se muestra como una alternativa a SAP, con la principal funcionalidad de ser la competencia a ERP.

Añade distintos módulos, entre los que encontramos, Gestión y compraventa, CRM, Gestión de proyectos, Sistema 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 y licencia correspondiente a una serie de módulos bajo AGPL.

Como podemos comprobar, OpenERP es el ERP OpenSource que cuenta con más módulos libres para poder añadirlos.

Presenta una **arquitectura basada en cliente- servidor**. El **servidor** se desarrolla en lenguaje **Python** y va a ser utilizado por el desarrollador para conseguir llevar a cabo los distintos módulos. El **cliente**, va a hacer uso de servicios web (**XML- RPC**) para establecer la comunicación con el servidor.

Ya a partir de la versión seis, OpenERP ofrece la posibilidad de poder distribuirse a través de la nube.

### Conclusiones

Como hemos podido comprobar anteriormente, hoy en día, el software de gestión empresarial lo domina el mercado propietario, destacando tres compañías principales, siendo la más conocida SAP. Actualmente, estas compañías siguen desarrollando más productos que cada vez están más orientados hacia su entorno en la nube. Igualmente, han ido apareciendo más soluciones libres que se empiezan a utilizar cada vez más.

También es importante que señalemos una serie de problemas que presenta el mercado de propietario a los distintos educadores como puede ser la adquisición de software. En la mayoría de los casos, el software se puede configurar por los consultores empresariales ya que éstos realizan todo el trabajo, dejando poco espacio a los agentes externos. Generalmente, las empresas son las encargadas de instalar, configurar y enseñar el funcionamiento de un paquete, además de proporcionar los distintos mecanismos de aprendizaje del sistema que estén controlados por ellos.

#### ○ Elección de un software ERP

Una vez que llegue el momento de elegir un ERP, debemos tener en cuenta una serie de factores, prestando especial atención a:

1. Conseguir ofrecer soporte a todas las áreas que necesite la empresa
2. Que sea fácil de utilizar y rápido.
3. Que utilice los estándares ya establecidos.
4. Debe contar con un período de migración y adaptación fácil y rápido.
5. Que proporcione informes y análisis además de seguridad que implemente.



6. En lo referente al nivel técnico, debemos observar diferentes elementos, como pueden ser:

- La posibilidad de utilizar hardware y software que posee la empresa.
- El sistema operativo necesario para poder ejecutarse.
- Las bases de datos obligatorias que son necesarias a la hora de la implantación de la plataforma.
- La experiencia y nivel de implantación de la plataforma.

Por último, debemos atender una serie de cuestiones diferentes como pueden ser el método y tiempo que se necesitan en la formación, el tipo de garantía correspondiente al producto, el ciclo de vida, la existencia de nuevos servicios de soporte y las diferentes posibilidades de existan para el mantenimiento.

Aunque no es tarea fácil el seleccionar una plataforma ERP, es conveniente ofrecer una respuesta al planteamiento del software ERP- CRM que vamos a llevar a cabo para el desarrollo de esta unidad formativa, de tal forma que seleccionaremos un software libre por todas las ventajas que ya hemos detallado.

De todas formas, vamos a realizar una comparativa entre los dos paquetes más importantes que hemos estudiado.

		
TECNOLOGÍA	Java and Javascript SQL and PL/SQL XML XHTML	Python SQL and PL/SQL XML
ARQUITECTURA	WAD Application MDD Dictionary MVC	MVC PostgreSQL database server Application server Open Object client-web
LICENCIA	Mozilla con cláusulas	GPL
CLIENTE	Solo Web	Web y aplicación Desktop
RENDIMIENTO	Menor	Mayor
PERSONALIZACIÓN	Complicada	Sencilla

- **Ventajas y desventajas de los sistemas ERP**

### **Ventajas**

- Posibilidad de que el sistema se adapte a nuestras necesidades.
- Consistencia de datos y acceso a la hora de tomar decisiones.
- Actualización de la información empresarial a tiempo real.
- Eliminación de datos redundantes.
- Eficiencia empresarial.
- Control de la actividad empresarial.
- Integración de todos los elementos (internos y externos) que tengan relación con la empresa.
- Disminución del tiempo de vida.

### **Desventajas**

- Costoso tanto en tiempo como en dinero.
- Es necesario tener nuevos aprendizajes.
- Ante el cambio de una estructura empresarial, es necesario modificar su ERP.
- Dificultades de integración de algunos productos.
- Ante un posible fallo en el sistema, volver atrás es un proceso bastante costoso.
- Desconfianza de los empleados ante un cambio de metodología.

## **1.3. Concepto de CRM (sistemas de gestión de relaciones con clientes).**

Los **CRM** (*Customer Relationship Management*) se refieren a la respuesta que un sistema de información puede ofrecer a todos los requisitos requeridos por una determinada empresa en relación a sus clientes.

De todas formas, debemos señalar que **los CRM no son sólo un sistema de gestión empresarial, sino que también se refieren a la visión que puede tener un cliente dentro de una empresa**. El cliente es la parte central mientras que todos los procesos van a ir destinados a que este consiga mejorar su relación con la empresa.

Los sistemas CRM incorporan los mecanismos necesarios para poder establecer relaciones que sean duraderas y a su vez, satisfactorias con los distintos clientes.

Para conseguir llevar a cabo este proceso, van a intervenir distintos departamentos de la empresa como pueden ser, el de ventas, marketing y atención al cliente ya que son los que están en constante contacto con el comprador.

**Los sistemas CRM se caracterizan, sobre todo por tener dos partes bien diferenciadas; una que se va a encargar de la lógica operacional y, otra que va a tener como función, analizar la información disponible (lógica analítica).**

Mediante la **lógica operacional** podemos llegar a los procesos de:

- Automatización de ventas, productos y clientes para tener la información organizada.
- Automatización de márketing para gestionar las diferentes campañas.
- Gestión de soporte en la empresa.
- Gestión de servicio de atención al cliente.
- Organización del call Center.
- Métricas que determinen el funcionamiento del negocio.

Y, mediante la **lógica analítica**, podremos:

- Analizar la información existente para poder llevar a cabo las diferentes campañas de márketing.
- Indicadores que nos muestren el estado de la empresa.
- Modificación de estrategias según los cambios realizados.

## 1.4. Revisión de los CRM actuales.

Podemos dividir los CRM actuales en **dos categorías**:

- Los que están **integrados en un sistema ERP**: Se refieren a aquellos CRM que pertenecen a los mismos propietarios que los ERP, como pueden ser, entre otros, el CRM de SAP, Oracle y Microsoft.



- Los que son **exclusivos**: Los exclusivos se refieren a los que están formados por todo tipo de proyectos, como pueden ser, OpenSource o sugarCRM. Estos se utilizan cada vez menos, por lo que pueden llegar a desaparecer algún día.

### 1.5. Concepto de *data warehouse* (Almacenes de datos).

**Data warehouse** es un almacén de datos que recopila toda la información correspondiente a una organización o una empresa determinada. Esta información va a ser de gran utilidad a la hora de tomar decisiones.

Lo podemos definir también como un expediente de una empresa que va a contener información transaccional y operacional. Esta información se va a almacenar en una base de datos diseñada específicamente para proporcionar análisis.

Es probable que estos almacenes de datos contengan gran cantidad de información que, en muchos casos, se van a dividir en pequeñas unidades lógicas, denominadas **centros comerciales**.

Las funciones principales de los Data Warehouse, las podemos resumir en:

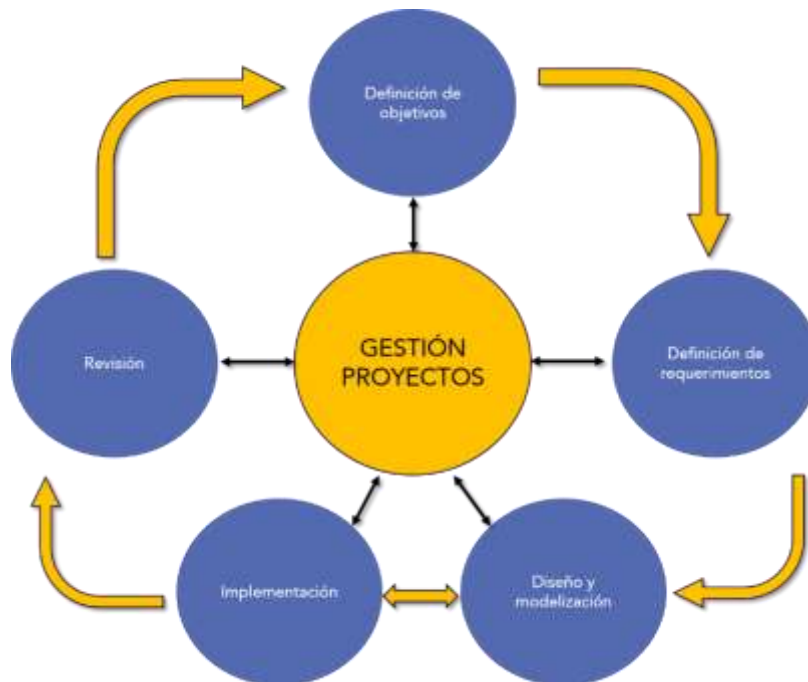
- Integra los datos de las correspondientes bases de datos distribuidas, facilitando una descripción global junto con un análisis comprensivo en el almacén de datos.
- Separa aquellos datos que se van a utilizar en las operaciones cotidianas en el almacén de datos, con el propósito de publicación y ayuda a la hora de tomar decisiones.

### 1.6. Revisión de los *data warehouse* actuales y posiblemente incorporados a los sistemas ERP- CRM.

Podemos diferenciar una serie de fases a la hora de realizar la implantación de un *data warehouse*.

Las distintas fases que vana formar este proceso, tienen como objetivo principal conseguir ser la herramienta más utilizada para sacar el máximo rendimiento de la aplicación.

Siguiendo los diferentes pasos de esta metodología, al iniciar el Data Warehouse por una determinada área, debemos conseguir los resultados más favorables en el menor espacio de tiempo.



*Metodología propuesta por SAS Institute: la "Rapid Warehousing Methodology"*

A continuación, vamos a detallar la **función principal** que se va a desarrollar en cada fase.

- **Definición de los objetivos:** donde vamos a declarar los objetivos necesarios para el desarrollo de un proyecto determinado.
- **Definición de requerimientos de información:** en la mayoría de proyectos, se debe contar con las técnicas más novedosas como son, en este caso, las que referencia data warehouse, encargándose de analizar las distintas necesidades para saber administrar las principales ventajas que este sistema puede aportar.
- **Diseño y modelización:** la fase anterior se va a encargar de proporcionar las principales bases para llevar a cabo el diseño y

modelización del data warehouse. Y esta, tiene como objetivo principal identificar las distintas fuentes de datos junto con las correspondientes transformaciones para, a partir de aquí, obtener el modelo lógico de datos del data warehouse.

Este modelo lo forman las diferentes entidades y relaciones que nos permiten resolver las necesidades específicas referentes al negocio de la organización.

Después, podemos traducir el modelo lógico de datos al físico para almacenarlos en el data warehouse y podrá definir la estructura de almacenamiento.

- **Implementación:** a la hora de implantar el data warehouse, debemos seguir una serie de pasos que, a continuación, detallamos:
  - Extraer datos del sistema y transformarlos.
  - Cargar datos ya validados en e.
  - Explotación data warehouse mediante distintas técnicas, como pueden ser, entre otras:
    - Query & Reporting
    - On-line analytical processing (OLAP)
    - Executive Information System (EIS) ó Información de gestión
    - Decision Support Systems (DSS)
    - Visualización de la información
    - Data Mining ó Minería de Datos.

Toda la información que necesitamos para poder llevar el control sobre los datos, la podemos almacenar en lo que denominamos los metadatos técnicos. Estos metadatos deben ser accesibles también por los usuarios finales y administradores para poder utilizarlos o modificarlos.

- **Revisión:** una vez que ya tenemos instalado el data warehouse, no finalizamos nuestra tarea, podemos realizar una revisión del mismo planteándole una serie de preguntas para ir actualizando y potenciando su función y utilización.

- **Diseño de la estructura de cursos de formación:** una vez mantenidas las reuniones necesarias con los distintos usuarios, podemos comenzar a practicar sobre todo lo que llevamos desarrollado para ir afianzando los distintos conceptos que nos van a servir para ir formando a los usuarios.

## 1.7. Sistemas gestores de bases de datos compatibles con el software.

Hasta ahora hemos visto distintas herramientas para cumplir las necesidades empresariales de analizar la información recopilada y obtener conocimiento de los datos (ERP o CRM). Seguramente el inconveniente de estos sistemas son los tiempos de respuestas de las transacciones ejecutada ya la información está repartida en diferentes módulos, de esta forma se diiculta el proceso del análisis de la información asé como los reportes y demás.

Como solución de esta problemática, surgieron los **Datawarehouse o Almacenes de Datos**. Se basa en una base de datos central y donde todas las aplicaciones hacen llamadas a ella. Está implementada para soportar todo tipos de herramientas de análisis.



La principal ventaja de estas herramientas es la organización a la hora de almacenar la información, ya que lo realiza de una forma homogénea y fiable. Realiza un análisis multidimensional para ver desde distintos puntos de vista todas las dimensiones del negocio.

A la hora de construir una Data WareHouse debemos de contar con las herramientas que responda a las necesidades planteadas y detectadas en el sistema.

Por un lado, a este tipo de sistema acceden una cantidad pequeña de usuarios pero con una gran necesidad de información. Las consultas ejecutadas en el dicho almacén de datos van a ser complejas y contando con una gran cantidad de información, por esta razón las máquinas donde se aloje el sistema deben contar con unas **altas prestaciones** en todos sus componentes.

Otra de las características que deben presentar estas máquinas es la **escalabilidad** para dar soporte a los nuevos elementos que tendremos que instalar, a consecuencia de todo esto, tendremos que apostar por una arquitectura abierta para la estructura del equipo.

Con respecto al software a utilizar, el **Sistema de Gestión de Base de Datos (SGBD)** es otro de los elementos más importantes del sistema. Independientemente de la información almacenada, se deberá de utilizar un SGBD que utilice una técnica de Base de Datos relaciones o multidimensionales, de esta forma, permite a las tablas poder estar relacionadas entre sí y por tanto podemos sacar mayor provecho a la información contenida en ellas.



Aunque la mejor opción son las **Bases de Datos Multidimensionales** por las mismas razones por las que elegimos las características físicas de las máquinas: escalabilidad, prestaciones y consolidación.

Este tipo de base de datos post-relacionales abre un mayor abanico de posibilidades y dan soluciones a las limitaciones que presenta las bases de datos relacionales ya que con menos recursos hardware, es capaz de ejecutar consultas mucho más complejas y que trabaje con mayor cantidad de información.

A continuación, detallaremos la extracción y manipulación de los datos. Para realizar este tratamiento de la información es imprescindible contar con las herramientas que permitan controlar y automatizar las necesidades de los *Data Warehouse*.

Estas **herramientas** deben proporcionar las siguientes **funciones**:

- Control en el tiempo de respuesta
- Acceso a diferentes tecnologías tanto a nivel de hardware como de software.
- Gestión integrada de la extracción, transformación y carga del Data Warehouse.
- Manejo de excepciones y archivos logs

## 1.8. Verificación de la instalación y configuración de los sistemas operativos y de gestión de datos.

Una vez que hemos seleccionado el software correspondiente, en este caso **OpenERP**, existen una serie de requisitos previos que debemos tener en cuenta antes de llevar a cabo su instalación, como:

- Soporta los sistemas operativos Linux y Windows.
  - **Linux**. Ofrece paquetes Debian oficiales que se pueden descargar e instalar de forma tradicional. También cuenta con paquetes *redHat* no oficiales.
  - **Windows**. Ofrece la posibilidad de poderlo descargar mediante un ejecutable autoinstalable.
- Los distintos clientes cuentan con la posibilidad de poder instalarlo en sistemas operativos Windows, Linux e iOs.
- Necesita una serie de requerimientos hardware que vana a depender de nuestra empresa junto con los distitnos usuarios del sistema.
- En referencia a la base de datos que debemos instalar, seleccionaremos *PostgreSQL* por ser una versión bastante recomendable.

### Instalación de la base de datos

1. Lo primero que debemos hacer cuando deseemos instalar la base de datos, es **obtener el software correspondiente** en la dirección que tenga asociada, como puede ser:

<http://www.postgresql.org/download/>

2. Si nos disponemos de un sistema Windows, podemos **ejecutar** el fichero directamente.
3. **En caso de tener Debian** como sistema, debemos conectarnos a Internet como supervisor, de la siguiente forma:
  - a. Primero, ejecutaremos *apt-get install postgresql*
  - b. Y, a continuación, instalaremos el gestor de la base de datos:
  - c. *apt-get install pgadmin3*
4. Por último, realizaremos las **pruebas oportunas para comprobar su correcto funcionamiento**.

### Pruebas de la base de datos

1. En primer lugar, comenzaremos abriendo el cliente gráfico **PgAdmin III**.
2. A continuación, debemos **comprobar la base de datos**.
3. Seguidamente, ya podemos **crear una nueva tabla**.
4. En esta nueva tabla que nos hemos creado, ya podemos **añadir los datos correspondientes**.
5. También disponemos de la opción de **buscar determinados elementos** de la tabla.
6. Realizaremos las **modificaciones** que deseemos.
7. En este apartado, ya podemos **eliminar la tabla**.
8. Nos **creamos una nueva tabla**.
9. **Importamos los datos** en la tabla.
10. **Listamos todos los datos** existentes.
11. **Eliminamos** la tabla.
12. Podemos **repetir todos los pasos** desde el cliente de texto psql (SQL Shell).

### Pruebas para el sistema operativo

- Desde el ordenador personal, podemos seguir realizando los siguientes pasos:
  1. Seleccionamos un navegador.
  2. Para conectarnos a una determinada página de Internet.
  3. Hacemos un ping sobre la dirección de la tarjeta instalada.



4. Realizaremos un Telnet al puerto 5432 de la dirección de la tarjeta instalada.
5. Hacemos ping a la dirección localhost (127.0.0.1).
6. Y, realizamos un telnet al puerto 5432 de la dirección localhost.
7. Podemos realizar un ping al nombre DNS de nuestro ordenador.
8. Por último, realizamos un ping a la dirección de la tarjeta instalada desde un ordenador conectado a nuestra red.

- **Desde otros ordenadores:**

1. Primero hacemos un ping al nombre de nuestro ordenador desde otro que pertenezca a nuestra red.
2. Cuando tengamos que hacer el acceso desde otras redes, repetiremos los dos puntos anteriores desde distintos ordenadores de otras redes diferentes.

## 2. Instalación y configuración de sistemas ERP- CRM

### 2.1. Tipo de licencia.

Siempre que trabajemos con algún tipo de productos industriales, debe existir un contrato entre el creador y el comprador, que denominaremos licencia.

Por tanto, la licencia, es un documento que detalla una serie de obligaciones que debe tener el desarrollador con el usuario en cuestión y, al contrario. Mediante el tipo de licencia, se van a indicar las posibilidades que existen para su utilización, modificación y distribución.

Hoy en día, podemos encontrar un gran número de licencias y, cada una de ellas, va a tener unas características diferentes.

Una clasificación de las licencias, la podríamos llevar a cabo dividiéndolas en:

- **Licencias de código abierto.** Cuando podemos hacer uso de un determinado producto sin muchas restricciones.  
De la misma forma, las licencias de código abierto, se van a dividir también en:
  - **Licencias permisivas.** No tienen limitación a la hora de utilizarla.  
BSD, MIT, Apache
  - **Licencias robustas.** Presentan distintas limitaciones que debemos tener en cuenta antes de su utilización.
    - Fuertes. Se debe distribuir de la misma forma que la original. (GNU, GPL, Eclipse)
    - Débiles. Se distribuye de forma muy parecida a la original, aunque pueden tener una licencia diferente. (Mozilla, Open Source)
- **Licencias de código cerrado.** A la hora de utilizar un producto, debemos tener en cuenta un gran número de restricciones. (EULA, y CUPS).

Una vez estudiados los diferentes tipos de licencias, podemos pasar a diferenciar entre los diferentes tipos de software, que los podemos clasificar atendiendo a diferentes **características**, como pueden ser:

- Tipo de licencia
- Coste

Estos parámetros van a ser algunos de los principales aspectos a tener en cuenta a la hora de seleccionar los diferentes tipos para una empresa.

Lo mismo sucede con la clasificación que podemos hacer atendiendo al **tipo de licencia** con la que se puede distribuir.

- Encontramos **software de código abierto** que permite su comercialización bajo una licencia de código abierto, permitiendo acceder al código fuente, modificarlo y adaptarlo a las distintas necesidades que determine la empresa.
- **Software propietario** que va a depender de las distintas características que presente la licencia.

Y si tenemos en cuenta el **coste**, podemos diferenciar entre:

- **Freeware**. No tiene coste.
- **Payware**. Necesita realizar algún tipo de coste.
- **Shareware**. Se distribuye sin coste, aunque sólo para utilizarlo durante un período de tiempo determinado.

## 2.2. Tipo de instalación. Monopuesto. Cliente/ Servidor.

Antes de comenzar con un proceso de instalación, debemos comprobar siempre que cumple con una serie de requisitos necesarios para la instalación que se deben cumplir antes de comenzar con el proceso, como pueden ser:

- **Monopuesto**

Realiza la instalación del software en un ordenador, de tal forma que sólo va a tener acceso al sistema el propietario del ordenador.

Esta instalación es conveniente para entornos de desarrollo o aprendizaje, No para entornos productivos.

### ○ **Cliente- Servidor**

En este caso, vamos a instalar todos los datos y la gestión de éstos en un ordenador, de tal forma que podemos acceder al control y administración desde distintos puntos de la organización, instalando una aplicación (cliente) con la que nos vamos a poder conectar al ordenador que posee los datos (Servidor).

Es una configuración bastante flexible ya que permite acceder a la aplicación desde cualquier terminal.

Es conveniente utilizar la configuración cliente- servidor para desarrollar diferentes tareas de administración, amntenimiento, ampliación, etc. en el servidor de forma eficiente.

Aunque es un tipo de configuración muy ventajosa, también debemos señalar algún tipo de inconveniente que presenta, como puede ser la infraestructura de comunicaciones que plantea, en concreto, el fallo que presenta. Por lo que tendremos que ser muy cuidadosos si no queremos provocar un fallo en el funcionamiento de una empresa.

### ○ **Cliente- Servidor Web**

Es una configuración bastante parecida a la anterior, con la salvedad de que, en este caso, utiliza una interfaz basada en estándares web. De esta forma, permite el acceso y la gestión, desde cualquier navegador que esté instalado en la máquina cliente.

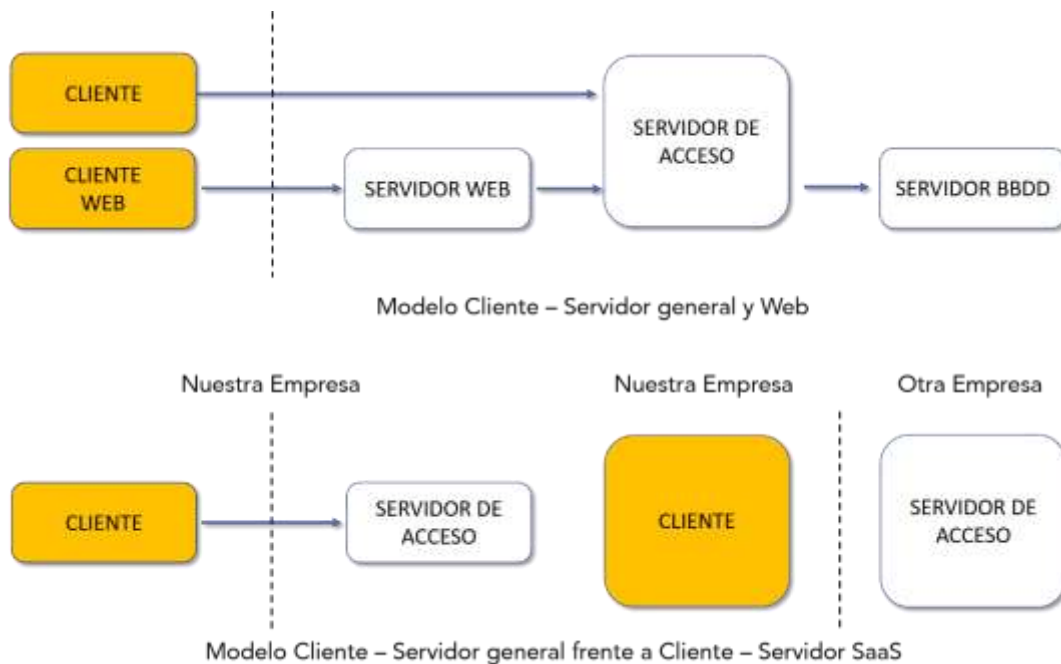
Esta configuración ofrece mucha más flexibilidad que las anteriores al poder hacer uso de cualquier navegador y no necesita ninguna aplicación antes de comenzar con el proceso de instalación.

### ○ **SaaS**

Es una configuración que utiliza la estructura cliente- servidor, pero, en este caso, el servidor se encuentre en una empresa externa que se va a encargar de la gestión y mantenimiento del software.

Para lograr un funcionamiento correcto, se contrata una empresa externa a la que se le contrata el servicio y, los datos, van a viajar hacia ella utilizando los medios de comunicación tanto públicos como privados.

Si bien es cierto que esta configuración dispone de unos precios ajustados y elimina la tarea de mantenimiento, también debemos comentar que, en este caso, perdemos el control del servidor ya que se encuentre fuera de nuestras infraestructuras.



## 2.3. Procesos de instalación del sistema ERP- CRM.

### ○ Requerimientos hardware y software de los ERP- CRM

Antes de comenzar con el proceso de instalación, debemos asegurarnos de cuál es el software ERP que más nos interesa para que el hardware funcione de la mejor forma posible. Debemos llevar a cabo un análisis riguroso a la hora de seleccionar un determinado software.

En este caso, **nos vamos a decantar por OpenERP**, ya que cuenta con unos requerimientos mínimos para su funcionamiento, destacando como el más importante la configuración de la base de datos.

Como hemos podido comprobar, no dispone de valores estándar para poder utilizar, por tanto, **podemos partir de una recomendación básica y realizar las correspondientes pruebas de funcionamiento con cargas reales**, para que, de esta forma, determinemos nosotros mismos los requerimientos más importantes.

Si hemos realizado una planificación correcta, podemos partir de un sistema base para, a continuación, ir ampliando según las necesidades que se vayan presentando.

Como ya hemos indicado, **uno de los principales factores del servidor OpenERP es la base de datos**, por lo que tenemos que ser muy cuidadosos a la hora de configurar de forma correcta PostgreSQL, especificando hasta los balanceos correspondientes de carga entre distintos servidores.

En cuanto a las **bases de datos**, debemos conocer que los **factores** más importantes son:

- El número de conexiones que se pueden realizar a la vez.
- La memoria.
- Espacio de almacenamiento.

Y, una vez que realizamos la instalación y configuración del servidor, podemos llevar a cabo las diferentes pruebas para comprobar el rendimiento mediante la herramienta **PGAdmin (Explain query)**. Así podremos comprobar los tiempos de acceso y ejecutar las consultas correspondientes siempre que las tablas estén correctamente definidas.

En la siguiente dirección, disponemos de una guía de mejora del rendimiento:

[http://wiki.postgresql.org/wiki/Mejorando\\_el\\_rendimiento\\_de\\_tu\\_Postgresql\\_Server](http://wiki.postgresql.org/wiki/Mejorando_el_rendimiento_de_tu_Postgresql_Server)

- **Instalación bajo Microsoft Windows.**

Lo primero que vamos a hacer es descargar los ficheros del servidor OpenERP (Server) y del servidor web (web Client) de la página oficial correspondiente.

<http://www.openerp.com/downloads>

A continuación, podemos instalar el servidor siguiendo una serie de pasos:

1. Debemos **crearnos un usuario** bajo PostgreSQL para OpenERP asignándole privilegios de administrador asignándole una clave. Podemos hacer uso de la herramienta PGAdmin III.
2. Podemos **lanzar el programa de instalación**. El proceso no es muy complicado y no debemos olvidarnos de establecer los diferentes

- datos en el servidor PostgreSQL: nombre o dirección IP, puerto, usuario y clave de acceso.
3. **Comprobaremos que los puertos NET- RPC y XML- RPC esten abiertos** desde la consola de Windows, mediante el comando: **netstat /a**.
  4. Efectivamente, tras escribir el comando, podemos observar que los puertos están abiertos. Además, debemos **comprobar si el servicio se levanta de forma manual o automática**. Cuando se trata de entornos de producción, siempre se hace de forma automática.
  5. Para poder acceder al servidor desde el exterior, necesitamos **configurar el cortafuegos**. Siempre es habitual, que se deniegue el acceso a la máquina por os que, en ese caso, abriremos o desactivaremos, de forma manual los puertos.
  6. Podemos comprobar si el fichero **openerp- server.conf** existe en el servidor.
  7. En este momento, ya tenemos el servidor configurado y se debe estar ejecutando. Por tanto, podemos **instalar un cliente en la máquina para llevar a cabo las distintas pruebas de conexión**. Podemos instalar el cliente, descargando el software de la página oficial **GTK Client**.

Una vez que ya hemos conseguido establecer la conexión, podemos realizar una pequeña prueba para comprobar OpenERP omitiendo los asistentes y así, crearnos la base de datos correspondiente. Para ello, nos conectaremos con el cliente y, una vez que salga el mensaje de base de datos no encontrada, sólo debemos omitirlo.

A continuación, abrimos el menú **Archivo/Bases de Datos/Nueva base de datos** y pasamos a realizar las configuraciones necesarias:

- Para **OpenERP**, la clave de superadministrador, es **admin** por defecto. Mediante esta clave, vamos a poder controlar todo el sistema si se produce algún fallo.
- En **OpenERP**, el nombre de la base de datos no es tan importante, ya que al final, se va a eliminar.
- Dejamos activada la carga de datos de prueba.
- El idioma viene por defecto.
- En **OpenERP**, la clave del administrador de esta base de datos, también es **admin**.



Es importante que omitamos el asistente cuando vayamos a realizar las diferentes pruebas de conexión.

1. Para continuar, podemos instalar **WebClient- Server** y seguimos con el asistente.
2. Debemos comprobar que el estado del **puerto 8080** esté abierto ya que es el que vamos a utilizar para poder conectarnos desde nuestros clientes web.
3. Podemos **añadir una excepción al cortafuegos para TCP en el puerto 8080**, para conseguir que sea accesible al exterior.
4. Llegados a este punto, ya podemos realizar alguna **prueba que verifique la conexión al puerto 8080** con el explorador junto con los dos usuarios que ha creado la instalación
  - El usuario **admin** que tiene como clave, **admin**.
  - El usuario **demo** que tiene como clave, **demo**.
  - Para el cliente de prueba, podemos utilizar cualquier tipo de navegador que soporte flash.
5. Por último, debemos **configurar las preferencias** que tenga cada usuario al idioma español (Preferences/Spanish) siempre y cuando éste, no venga seleccionado por defecto.

- **Instalación bajo Linux Ubuntu**

En el caso de Linux, lo primero que debemos hacer es crearnos un usuario de la base de datos para nuestro servidor **OpenERP** y asignarle los permisos administrativos además de establecer su contraseña.

1. Para crear un usuario desde Shell que tenga como login el usuario *postgres*, iniciaremos sesión con este usuario y, a continuación, llamamos al script **createuser** ya que es el que va a realizar esta tarea.

El *script* nos va a preguntar por el tipo de permisos que queremos asignarle al nuevo cliente, que como mínimo, debe ser capaz de crear bases de datos.

```
sudo su postgres  
createuser openerp -P
```

2. Una vez que hayamos establecido el usuario dentro del servidor de la base de datos, necesitamos crearnos un directorio home del mismo para determinar aquellos permisos necesarios.

```
sudo mkdir /home/openerp  
sudo chown openerp.nogroup /home/openerp/
```

Ya podemos descargar los ficheros necesarios para Ubuntu (si todavía no lo hemos hecho), de la página oficial,

<http://www.openerp.com/downloads>

3. Antes de comenzar la instalación del servidor, descargamos las librerías necesarias y las configuramos ya en nuestro sistema.

```
sudo apt-get install graphviz libdct4 libcgraph5 libgraph4  
libgvc5 libgvpr1 libpathplan4 libyaml-0-2 postgresql-client  
python-libxslt1 python-lxml python-psycopg2 python-pychart  
python-pydot python-pyparsing python-support python-tz  
python-yaml
```

4. Ya poder montar el servidor, mediante la orden

```
sudo dpkg -i openerp-server...deb
```

En el caso de que se produzca cualquier tipo de error, podemos intentar resolverlo mediante la instrucción

```
sudo apt-get install -f
```

5. Cuando instalamos Linux de PostgreSQL sólo nos va a permitir realizar conexiones vía sockets Unix. Por tanto, tenemos que habilitar el acceso por medio de vía TCP/IP y tenemos que modificar

```
local all all ident por, local all all md5
```

6. Una vez llegados a este punto, PostgreSQL ya puede recibir entradas locales desde TCP/IP, así que ya podemos configurar OpenERP con los datos de usuario que hemos visto en puntos anteriores. Sólo tenemos que abrir el fichero de configuración y seleccionar usuario y contraseña.

7. A continuación, reiniciamos el servidor

**sudo /etc/init.d/openerp-server resart**

- Y comprobamos, mediante la orden `netstat -a` que los puertos 8069, 8070, 8071 y 5432 estan abiertos.
- Podemos comprobar el estado en el que se encuentra el cortafuegos. Si necesitamos establecer alguna regla, podemos hacerlo mediante la orden **sudo iptables -L**
- Ya, por último, nos podemos conectar a OpenERP para crear la nueva base de datos para el sistema, de la misma forma que lo hicimos para Windows.
- Como ya tenemos instalado el servidor, ahora podemos proporcionar la forma de acceder de los clientes web. Para ello, instamamos el servidor web Client descargado.
  - Es conveniente que, antes de comenzar, paremos el servicio OpenERP por seguridad.  
**sudo /etc/init.d/openerp-server stop**
  - Podemos instalar las dependencias que necesitamos  
**sudo apt-get install python python-dev build-essential**  
**sudo apt-get install python-setuptools**
  - Y ya con el entorno preparado, podemos iniciar el proceso de instalación. Una vez que finalice el proceso de ejecución, podemos comprobar que el servidor tiene un correcto funcionamiento, mediante la orden  
**openerp-web**
- Llegados a este punto, ya tenemos el servidor web instalado aunque no se arranca de forma automática. Así que tenemos que modificar el sistema de arranque.

- Podemos copiar el fichero de configuración del servidor web en /etc
- Y escribimos el correspondiente script de arranque que denominamos openerp-web y lo podemos poner en /etc/init.d con su correspondiente contenido.
- Para crear las ligaduras de arranque de los directorios /etc/rc2.d /etc/rc3.d /etc/rc4.d /etc/rc5.d y las de parada en /etc/rc1.d y /etc/rc6.d lo podemos hacer a mano o de forma automática, mediante las instrucciones:
 

```
# chown root: /etc/init.d/openerp-web
# chmod 755 /etc/init.d/openerp-web
# update-rc.d openerp-web defaults
```
- Para finalizar podemos ejecutar los comandos lanzando los servidores
 

```
# service openerp-server start
# service openerp-web start
```
- Llegados a este punto, ya nos podemos conectar para realizar las pruebas necesarias.

Ya hemos completado el proceso de instalación del sistema OpenERP bajo distribución Linux. Sólo nos faltaría reiniciar el equipo para comprobar que todo arranca correctamente.

Este proceso es un poco más complicado que el que seguimos en Windows aunque, con los dos, obtenemos el mismo resultado.

## 2.4. Módulos, Gestión y Actualización de un sistema ERP- CRM.

### Gestión de usuarios

Entre las funciones más importantes en la gestión de usuarios, podemos encontrar la creación, mantenimiento y borrado de los distintos usuarios pertenecientes al sistema además de sus correspondientes datos.

Es una tarea muy importante, sobre todo por desarrollar:

- **Gestión eficiente del sistema.** La persona que se quiere realacionar con él, debe tener acceso a través de un determinado mecanismo.
- **Seguridad.** Ya que va a tener el control del usuario que se conecta y la forma de hacerlo.

Para ver estos conceptos de forma más práctica, vamos a crearnos un usuario para asignarle un determinado grupo, modificar sus datos para, a continuación, eliminarlo.

- A la hora de crear un usuario, accedemos a la lista que encontramos en **Administración/ Usuarios**. Podemos acceder a los datos haciendo clic sobre ellos.
- Para crear un nuevo cliente, pulsamos el botón **Nuevo** de la barra de botones.
- De todos los usuarios existentes, abrimos uno para que podamos realizar las modificaciones pertinentes pulsando el botón **Edit** en la interfaz web. Cuando finalicemos, no podemos olvidarnos de guardar los cambios.
- Para gestionar los grupos por parte de los clientes, seleccionamos de la parte inferior derecha una lista que contiene todos los grupos a los que pertenece el usuario. Cada grupo dispone de unos determinados permisos para el sistema.

### Gestión de clientes

En **OpenERP** entendemos que un cliente es cualquier entidad que mantenga una relación con el sistema. De tal forma que, tanto los compradores, proveedores, como el personal de la empresa se pueden identificar como clientes del sistema.

Podemos acceder a comprobar el listado de los diferentes clientes, mediante el recurso **Ventas/ Libreta de direcciones/ Clientes**

Es muy parecido al interfaz de solicitudes. En este caso, se dispone de tres pestañas que se van a ir incrementando con los módulos que vayamos añadiendo.

Cuando deseemos visualizar todos los datos del cliente, debemos comprobar en la parte superior que está seleccionado el nombre de la empresa, el idioma junto con dos casillas de verificación.

Debemos asegurarnos de que aparezcan nombradas como cliente o bien como proveedor, para que se establezcan de forma correcta y el sistema pueda cargar las listas de selección correctamente.

De estas tres pestañas que ya hemos mencionado, la primera corresponde a la **General** y aquí podremos configurar diferentes datos del cliente como pueden ser el nombre, dirección, teléfono, etc.

En las otras dos, podemos guardar información de la relación que tengamos con el cliente (**Ventas y Compras**), además de un área libre para diferentes notas (**Notas**).

En la gestión de clientes, podemos encontrar alguna complicación que va a depender de la empresa en la que estemos implantando el sistema.

Podemos tener empresas con pocos clientes y otras con muchísimos, aunque para los dos casos, podemos dividir los clientes en categorías para poder facilitar la gestión en un menor tiempo.

A través de **Ventas/Configuración/Libreta de direcciones/Categorías** podemos acceder a la gestión en la que podemos llevar a cabo las modificaciones que creamos más adecuadas.

### Añadir funcionalidades

La instalación que hemos llevado a cabo al principio del tema que estamos tratando, muestra un entorno bastante simple con funcionalidades muy básicas. Por tanto, podemos añadir aquellos módulos que deseemos para conseguir un sistema más provechoso.

Por eso, en este apartado, vamos a ver la forma en la que podemos instalar y desinstalar los módulos en nuestro sistema.

Para poder hacer uso de un módulo, antes debemos actualizar la lista buscando aquellos módulos que tenemos descargados en nuestro sistema. No es recomendable que llevemos a cabo la instalación de los módulos ya que es un proceso bastante complejo. Lo haremos accediendo al ítem de **Administración/ Módulos/ Actualiar lista de módulos** y así podemos comprobar los módulos que podemos instalar y las modificaciones que podemos desarrollar.

- **Añadir un módulo**

Primero, seleccionaremos el icono de la flecha, como actualizable y, si ya está instalado, como instalable. Haremos uso de los iconos que se muestran al final de cada módulo.

- **Desinstalar un módulo**

Marcamos el icono que deseamos eliminar mediante el icono de una cruz.

Tanto las operaciones de añadir y eliminar módulos las podemos repetir tantas veces como deseemos y, los cambios realizados, se guardarán todos juntos cuando finalicemos.

- **Ejecutar modificaciones**

Cuando ya tengamos las distintas configuraciones realizadas, tenemos que ejecutarlas, por lo que seleccionaremos el elemento en **Administración/ Módulos/Aplicar actualizaciones programadas**

Una vez que finalice este proceso, se ejecutará de forma automática un asistente de configuración para los siguientes módulos si lo necesitan

Por último, indicar que, cuando instalamos un nuevo módulo, aparecen dos situaciones distintas:

1. En la primera, el componente que queremos instalar no depende de ningún otro, de tal forma que, el sistema no va a desempeñar más acciones.
2. En la segunda, el elemento que deseamos instalar, va a depender de un conjunto de módulos. En este caso, se debe configurar el sistema para que estos elementos también se instalen.

- **Características de los módulos funcionales**

OpenERP dispone de una estructura basada en diferentes módulos que se pueden descargar de la aplicación y se ponen en disposición del público mediante repositorios que podemos ordenar dependiendo de su versión o tipo de módulo.

Existe una gran cantidad de módulos que podemos encontrar en:

<http://apps/openerp.com>

Una vez que encontremos el módulo, podemos descargarlo copiando al subdirectorio **addons** del directorio raíz en la instalación del servidor para posteriormente, actualizar la lista de módulos y configurarlo.



Si durante este proceso se detecta que falta algún módulo, aparecerá un mensaje indicando el tipo de problema ocasionado. En este caso, la aplicación, que debe contar con un sistema de importación de módulos puede proceder a realizar una copia directa.

- **Relación de categorías existentes en el sistema**

Cuando ya hemos instalado el sistema, podemos tener acceso a los diferentes módulos de **Administración/Módulos**.

En este apartado vamos a ver la gestión de actualización de los módulos reconocidos, su instalación o desinstalación, la importación al sistema de módulos junto con la realización de diferentes tareas.

Añadir y eliminar módulos, lo podemos hacer en dos fases: primero, marcaremos las tareas desde el apartado **Módulos** y, a continuación, podemos pasar a realizar todas las funciones desde realización de las tareas pendientes.

El orden de los módulos puede ir determinado según el criterio del creador, pudiendo estar ordenados por nombre, fecha, autor, descargas, etc.

Podemos utilizar una caja de búsqueda para facilitar este proceso.

- **Módulos mínimos instalados**

- base: corresponde al módulo base del sistema.
- base\_setup: este módulo permite configurar el sistema cuando creamos una nueva base de datos.
- web\_livechat: actúa como soporte de aquellos que tienen un contrato de mantenimiento, añadiendo en la cabecera el botón de soporte.

- **Módulos disponibles tras una instalación**

Disponemos de una lista donde aparecen todos los módulos instalados en un sistema OpenERP después de su instalación. Estos módulos están siempre disponibles a pesar de que no tengamos acceso a Internet.

- **Módulos que tenemos que conocer**

Es conveniente que conozcamos una serie de módulos para conseguir un aprendizaje correcto y saber manejar el sistema. Para obtener más información, podemos acceder a la siguiente dirección:

<http://doc.openerp.com/v7.0>

- **account (contabilidad y finanzas)**

**Descripción.** Podemos utilizarlo para toda la contabilidad, como puede ser el análisis contable y de costes, contabilidad de terceros, gestión de presupuestos, impuestos, facturas de clientes y proveedores y, el estado de las cuentas bancarias.

- **Product (gestión de productos)**

**Descripción.** Nos va a permitir el manejo de diferentes productos y listas de precios en OpenERP, incluyendo variantes en el soporte de productos, distintos mecanismos de precios, información de proveedores, reservas (stock), unidades de medida, empaquetado y algunas propiedades individualizadas.

- **Purchase (compras)**

**Descripción.** Utilizado para la gestión de compras, permitiendo solicitar citas, crear facturas de proveedores, impresión de las distintas órdenes.

- **sale (ventas)**

**Descripción.** Es el módulo base para la gestión de citas y órdenes de ventas, además del flujo de trabajo con validación.

Puede incluir reserva-> Orden de venta-> Factura

Los diferentes tipos de facturación que puede utilizar son: factura en orden y en reparto y avance de factura (presupuesto).

Y, además, incorpora la gestión sobre la preferencia de socios, gestión de stocks y precios y otros métodos de reparto.

- ***crm (gestión de clientes)***

**Descripción.** Podemos utilizar este módulo para gestionar las diferentes relaciones con los clientes. Permite también gestionar tareas como pueden ser la comunicación, identificación, priorización, asignación, resolución y notificación. Cuenta con la opción de ir lanzando recordatorios de forma automática, eleva las peticiones, lanza métodos específicos entre otras opciones.

- ***hr (recursos humanos)***

**Descripción.** Utilizado para gestionar los recursos humanos, permitiendo crear puestos de trabajo y estructuras jerárquicas, hojas de trabajo, gestión fichaje entrada y salida. Cuenta con diferentes informes para la gestión.

- ***stock (gestión de almacenes)***

**Descripción.** Permite llevar a cabo la gestión de múltiples almacenes proporcionando distintos métodos de inventario, gestiona el valor del stock, maneja las distinciones de los métodos de inventario, gestión del valor del stock, etc.

- ***Marketing (campañas publicitarias)***

**Descripción.** Utilizado para llevar a cabo las diferentes campañas de marketing.

- ***account\_payment (pagos de contabilidad)***

**Descripción.** Proporciona una forma eficaz de manejar el pago de las distintas facturas y un mecanismo para automatizar los pagos.

- *delivery (reparto)*

**Descripción.** Ofrece la posibilidad de incluir otros métodos de reparto a las órdenes de venta y empaquetado.

- *point\_of\_sale (punto de venta)*

**Descripción.** Como principales ventajas, podemos destacar la creación directa de orden de compra, seleccionando el tipo de pago, el cálculo automático de la devolución de dinero, creación de factura y devolución de una venta.

- **Módulos para trabajar en España**

Podemos disponer de una gran cantidad de módulos que se instalan por defecto y que tiene traducción al castellano, aunque, también existen una serie de peculiaridades a la hora de gestionar una empresa de nuestro país, de tal forma que, debemos adaptar los módulos originales a la normativa actual.

Los módulos específicos para España, pueden ser, entre otros, los que observamos en el anexo adjunto.

## 2.5. Administración básica y configuración

- **Instalación de módulos**

Como venimos comentando a lo largo de esta Unidad Formativa, **OpenERP dispone de una gran cantidad de módulos con un gran número de aplicaciones que se han desarrollado por la comunidad.** Hoy en día, suele aparecer un nuevo módulo cada cuatro o cinco días, en las versiones más actuales.

Esto lo convierte en uno de los sistemas que más rápido está creciendo.

Aunque esta gran cantidad de módulos, también presenta el inconveniente de que el tiempo estimado en encontrar un módulo, sea cada vez más elevado.

Cuando ya encontramos el software y o descargamos, tenemos que incorporarlo al sistema. Este proceso no presenta demasiada dificultad y lo podemos llevar a cabo a través del menú **Administración** en el apartado de

**Módulos** podemos ir añadiendo cualquier tipo de aplicación que se encuentre en el directorio, simplemente haciendo clic en el elemento **Import Module**. Mediante este mecanismo, el fichero descargado se va a copiar al directorio **addons** del servidor.

Algunas veces, puede que este proceso falle. Si esto ocurre, es conveniente que traslademos los ficheros comprimidos de forma manual al directorio.

Independientemente de la forma en la que copiemos los datos, todavía no podemos utilizar el módulo ya que antes debemos incorporarlo a la base de datos mediante **Actualizar la lista de módulos** debiendo aparecer como uno más dentro de **Módulos**.

Llegados a este punto, lo que hemos hecho hasta el momento ha sido programar el sistema para que pueda llevar a cabo la instalación o actualización, aunque, estas funciones no se van a realizar hasta que no pulsemos **Aplicar actualizaciones programadas**.

Si nos encontramos el caso en el que el instalador ha detectado alguna dependencia que no se cumple, se nos informará de ello y tendremos que añadir el módulo en cuestión antes de continuar.

- **Configuración de módulos**

Podemos configurar los diferentes módulos mediante el asistente que tengamos cuando realicemos el proceso de instalación, en el menú **Configuración**. Estas opciones van a ser distintas dependiendo de cada módulo que tengamos que configurar.

- **Salvaguardar la configuración**

Una vez que ya tengamos el sistema configurado, es conveniente que realicemos una copia de seguridad de su configuración para no perder los cambios realizados. Podemos hacerlo para cualquier módulo, sólo instalando el módulo **base\_module\_record** y, a continuación, lanzaremos el asistente desde **Administración/ Personalización/ Creación de módulo/ Exportar personalizaciones como módulo**.

Para finalizar sólo tenemos que importar el módulo para poder instalarlo en el sistema.

- **Configuración del usuario**

En OpenERP podemos realizar bastantes configuraciones, como pueden ser los menús, su organización, los informes, las distintas ventanas o vistas, etc.

A continuación, vamos a ver algunas de las configuraciones más frecuentes.

- **Menús**

- Padre: cuando no dependen de nadie.
- Hijo: dependen de su padre.

Podemos acceder a la configuración de los menús desde **Administración/ Personalización/ Interfaz de Usuario/ Elementos de menú** y una vez seleccionado el menú a modificar, podemos acceder a sus propiedades.

Las principales opciones que podemos realizar dentro de un menú, pueden ser:

1. Seleccionar el orden en el que se va a visualizar.
2. Decidir el orden de las funciones a desarrollar, mediante el menú **Acción**
3. Configurar los iconos para la web, tanto para navegador móvil como para el escritorio.

- **Página principal y menú principal**

Mediante los menús **Acción Inicial** y **Acción de Menú** podemos personalizar de forma sencilla la página principal de cualquier usuario.

El primero nos ofrece la posibilidad de modificar la ventana que se puede visualizar cuando el usuario se conecta por primera vez mientras que el segundo, permite modificar en Formulario/Menú del cliente GTK.

- **Valores por defecto en los campos**

Por ser los administradores del sistema, tenemos la opción de poder fijar como predeterminado cualquier tipo de formulario. De esta forma, una vez que ya sea predeterminado, podemos seleccionar si queremos que aparezca de forma automática algún valor o si preferimos asignarle un valor a un campo para que siempre tenga el mismo dato.

- **Traducciones**

Existe la posibilidad de que el sistema traduzca, de forma automática cualquier campo correspondiente a un formulario asignándole un icono de traducción.

- **Gestión de los permisos**

Los sistemas OpenERP permiten gestionar los permisos de acceso mediante dos elementos principales, que son los usuarios y los grupos. Cualquier persona que desee interactuar con la aplicación, se debe autenticar mediante un usuario **Administración/ Usuarios/Usuarios** (dado de alta) que se corresponda con algún empleado de la empresa **Recursos Humanos/ Empleados**

En el usuario del sistema tenemos la opción de poder configurar los datos correspondientes al nombre del usuario, clave, correo electrónico, etc. Es posible que cada usuario pertenezca a un grupo determinado o a varios teniendo cada uno los privilegios de la cuenta en cuestión.

Cada grupo se puede expresar en:

- **Menús** a los que se puede acceder. Indican los menús que podemos utilizar.
- **Acceso a las tablas de la base de datos** (objetos). Nos ofrece información sobre lo que podemos desarrollar en los menús.

Una de las funciones más importantes de un administrador puede ser el definir los diferentes grupos. Por tanto, los grupos deben contar con una estructura representativa con capacidad para flexibilizarse.

Por ejemplo, una o varias personas pueden tener distintas tareas y, cada una de ellas debe estar representada por otro grupo distinto, asignándole a cada usuario los grupos que sean necesarios. De esta forma conseguimos un diseño flexible de fácil gestión.

A la hora de crearnos un grupo, podemos hacerlo mediante **Administración/ Usuarios / Grupos** y desde ahí podemos asignar o modificar los permisos necesarios a cada grupo.

- **Grupos que se crean por defecto**

Como nos encontramos ante un sistema altamente configurable, los grupos por defecto van a depender de los módulos que hemos instalado.

Para el desarrollo de esta unidad formativa, nos vamos a centrar en los grupos bases existentes y los diferentes menús que ofrecen acceso.

*Todos los tipos los detallamos en el anexo 2.*

- **Gestión de la base de datos**

Los datos son una de las principales herramientas de los sistemas de información. Una vez que ya tenemos instalado el sistema, la principal función del administrador es salvaguardar los datos de una determinada empresa.

Podemos gestionar los datos y las bases de datos gracias a las herramientas que incluyen los clientes para no tener que trabajar con el Servidor PostgreSQL de forma directa. Desde el cliente web, podemos acceder a las diferentes utilidades del botón **Bases de datos**.

Además, también contamos con la opción de poder realizar una copia de seguridad de cualquier base de datos. Para lograrlo, debemos proporcionar la contraseña de administrador.

**Si disponemos de más de una base de datos, es conveniente que la eliminemos antes de proceder a la copia de seguridad.**

De la misma forma que podemos realizar una copia de seguridad, también contamos con la posibilidad de poder restaurar desde un fichero creado a una nueva, configurando el nombre de la nueva base de datos y su contraseña.

Para todas estas opciones, podemos utilizar la herramienta PGAdmin en PostgreSQL que nos ofrece una serie de facilidades para crear y restaurar copias de seguridad.

- **Actualización de OpenERP**

OpenERP permite ir liberando las diferentes actualiaciones de forma mensual. Es bastante recomendable para aquellos sistemas que se encuentran en producción ya que no incluye capacidades nuevas entre versiones.

En cuanto a su funcionamiento, podemos llevarlo a cabo mediante dos pasos: primero sobrescribiremos el código del servidor con el de la actualización



para, a continuación, pasar a sincronizar los datos en la base de datos. En OpenERP, lo realizaremos mediante una serie de pasos que, a continuación, indicamos:

1. Realizaremos una copia de seguridad completa de todas las bases de datos y ficheros que se encuentren en el servidor.
2. Servidor OpenERP y Web.
3. Actualizamos a la última versión o, podemos instalar una nueva versión sobre la existente.
4. Ejecutaremos desde la línea de comandos:
  - `openerp-server.exe -d DB_NAME -u all`
  - `opener-server.py -d DB_NAME --update=all`

Para actualizar todos los datos de los módulos.

1. El servidor tiene que completar su arranque y, a continuación, parar el servidor mediante **Ctrl+C**. Podemos repetir este proceso para las distintas bases de datos que utilicemos.
2. Reiniciar todos los servicios.
3. Reinstalamos los clientes.

## UF2. Sistemas ERP- CRM. Explotación y adecuación.

### 1. Organización y consulta de la información.

#### 1.1. Introducción a las bases de datos.

Sobre los **años 70**, se fueron introduciendo en las empresas los diferentes sistemas gestores de base de datos, ofreciendo mejorar la gestión y organización del proceso productivo. Debido a esta dependencia que se crea, es conveniente que entendamos el sistema de una aplicación ERP- CRM para, de esta forma, poder adaptarla a cualquier proyecto de forma correcta.

Por tanto, es conveniente que recordemos algunos de los aspectos más importantes de las bases de datos, como pueden ser sus principales conceptos y procedimientos.

**Hoy en día, la mayoría de los sistemas cuentan con su correspondiente base de datos implantada (Oracle, PostgreSQL, MySQL, etc), basadas en los principios de las bases de datos relacionales.**

El **modelo relacional** permite estructurar los diferentes datos en tablas organizadas en filas y columnas, donde las filas van a ser las diferentes relaciones entre los conjuntos y valores y las columnas, los atributos correspondientes a esa relación, es decir, hacen referencia a un conjunto de información relacionada entre sí, almacenada de forma ordenada y con una estructura determinada.

Los sistemas relacionales deben cumplir una serie de **propiedades básicas** como pueden ser, entre otras, las que indicamos a continuación:

- Debe ser un sistema capaz de **eliminar la redundancia** entre sus datos para poder asegurar la coherencia entre todos (**integridad**).
- Proporcionar la posibilidad de **compartirlos entre varios usuarios** a la misma vez (compartición).
- Controlar el acceso y la configuración según unas reglas establecidas (**seguridad**).
- Asegurar que los datos pueden permanecer a lo largo del tiempo (**protección contra fallos**).

- Mostrar a los usuarios dependiendo de las necesidades de cada uno (**vistas**) utilizando algún tipo de lenguaje estructurado (interfaz de alto nivel: SQL).

A la hora de comenzar una base de datos, es necesario que conozcamos una serie de conceptos fundamentales, entre los que desatacamos los siguientes:

- **Dato.** Unidad mínima de información que puede almacenar un ordenador. Un dato se corresponde con una información, no con un tamaño.
- **Campo o atributo.** Unidad mínima de información que forma parte de una fila o un registro de la base de datos. Cada campo, puede almacenar un dato.
- **Fila o registro.** Conjunto mínimo de atributos relacionados que se pueden almacenar en una tabla. El mínimo número de campos que puede tener un registro es de uno.
- **Tabla.** Es la mínima unidad de estructura de una base de datos. Conjunto de registros o filas relacionadas que se almacenan en una base de datos bajo un mismo nombre.  
Todos los registros se almacenan en tablas y, las tablas, deben depender unas de otras a través de las relaciones.
- **Vista.** Se van a utilizar para mostrar un conjunto de información perteneciente a diferentes tablas.
- **Relación.** Hace referencia a las distintas condiciones que deben cumplir los datos para almacenarse en un determinado campo de una fila. Entendemos por relación, dos tipos de conceptos:
  - Por un lado, se refieren a las diferentes condiciones que debe cumplir un dato para poder almacenarse en un campo (como puede ser que la clave primaria sea un entero, mayor que cien, etc.)
  - Y por otro, las restricciones que debe cumplir un dato en un determinado campo (claves foráneas).

En los dos casos, las reglas deben marcar qué tipo de dato se va a insertar en el correspondiente atributo.

- **Consulta.** Las podemos utilizar para almacenar información de la base de datos. Son un conjunto de filas, que pueden tener una o varias

tablas solicitadas por el usuario dependiendo de distintos criterios de selección y ordenación.

Una consulta implica un control de acceso a los datos (**permisos**) y, control de acceso concurrente (**simultáneo**) para así, evitar las inconsistencias.

- **Informe.** Consulta que podemos realizar a la base de datos y se presenta de forma sencilla de analizar e imprimir.
- **Formularios.** Hacen referencia a las distintas pantallas creadas para manejar los diferentes datos simplificando de esta forma la gestión de la información. Normalmente están basadas en las vistas.
- **Procedimientos almacenados.** Diferentes órdenes que pueden realizar una tarea simple que se ejecuta y almacena en la base de datos.

Podemos estructurar el diseño de las bases de datos relacionales en tres niveles diferentes:

- **Nivel físico:** encargado del almacenamiento en un soporte, teniendo en cuenta, a nivel de bytes y ficheros que no esten relacionados entre ellos.  
Es un nivel muy utilizado para ficheros, índices sobre ficheros, sistemas de almacenamiento, uso y estructura.
- **Nivel lógico o conceptual:** permite crear una estructura en los datos mediante tablas y restricciones.  
Este nivel lo ven a utilizar las tablas, los campos de las tablas, restricciones de integridad, claves primarias, etc. Es decir, todo lo que se refiere el esquema relacional.
- **Nivel externo:** agrupa el nivel anterior en un conjunto de informaciones relacionadas sin importar la tabla a la que pertenezca cada una. De esta forma, ofrece al usuario un mejor y eficiente manejo.  
Utilizado por formularios e informes que se hayan creado mediante una o varias consultas.

Los niveles lógicos y conceptuales cuentan con un esquema único una vez que se hayan establecido. Almacenarán los datos siempre de la misma forma, el esquema relacional que implementemos en el nivel conceptual es muy raro que cambie.

Sin embargo, a nivel externo, podrán aparecer tantos esquemas como sean necesarios. Entendiendo por esquema al conjunto de vistas necesarias para que un usuario pueda realizar su trabajo de forma correcta.

Los diferentes procedimientos que podemos utilizar mediante la herramienta PgAdmin III a la hora de gestionar las bases de datos, pueden ser:

- **Creación de una base de datos**

Lo primero que debemos hacer una vez que nos conectemos al servidor, va a ser crear una base de datos, mediante un procedimiento sencillo. Seleccionaremos, del menú contextual, **Nueva Base de Datos**, estableciendo su correspondiente Nombre, propietario, codificación de caracteres, etc.

Una vez creada la base de datos, ya podemos añadir las distintas tablas según se hayan definido en el estudio entidad- relación junto con su posterior conversión al esquema relacional. Se deben agregar los campos necesarios a cada tabla además de las restricciones necesarias.

- **Modificación de una base de datos**

Dentro de la modificación de una base de datos, nos referimos a la inserción, modificación y borrado de la misma. Utilizaremos una herramienta no muy complicada de utilizar para cada procedimiento.

Partiendo del menú **Ver Datos**, seleccionaremos la opción que sea más parecida a las necesitadas requeridas. Cuando presentemos la vista, ya será más fácil:

- **Insertar** un nuevo registro, mediante fila libre.
- **Modificar** uno ya existente, seleccionando el botón de campo a variar del menú contextual.
- **Borrar** un registro utilizando también opciones del menú contextual.

Además, tenemos la opción de duplicar una fila haciendo uso de nuevo del menú contextual.

- **Creación de consultas y vistas**

Diseñar e implementar la base de datos en un sistema ERP- CRM, no va a ser nuestra tarea. Para ello, va a ser el sistema el encargado de configurar la base de datos de una forma idónea, las tablas con sus correspondientes cargos y las distintas restricciones.

Nuestra función principal va a ser saber entender y utilizar de forma correcta la estructura para que, de esta forma, podamos conseguir la información deseada.

Una de las formas más utilizadas para obtener información de una base de datos, es mediante el lenguaje SQL (a través de la sentencia SELECT), ofreciendo la información en forma de tabla. En caso de que conozcamos el lenguaje, su estructura y las restricciones, podremos utilizar sin ningún problema PostgreSQL. En el caso de no tener tantos conocimientos, lo llevaremos a cabo a través de la herramienta gráfica para la creación de consultas.

Podemos establecer una consulta, haciendo uso del menú contextual de la tabla, accediendo al elemento **Scripts/SELECT Script**.

Una vez que definamos la sentencia, mediante lenguaje SQL, podemos pasar a ejecutarla para obtener los resultados.

- **Procedimientos almacenados**

Es fundamental que mantengamos una buena seguridad cuando gestionemos las diferentes bases de datos, por tanto, se utiliza como mecanismo de seguridad y eficiencia la introducción en las bases de datos de los diferentes procedimientos almacenados. Simplemente son unas funciones con su correspondiente código que, pueden realizar alguna tarea determinada intentando siempre aumentar las capacidades de las bases de datos.

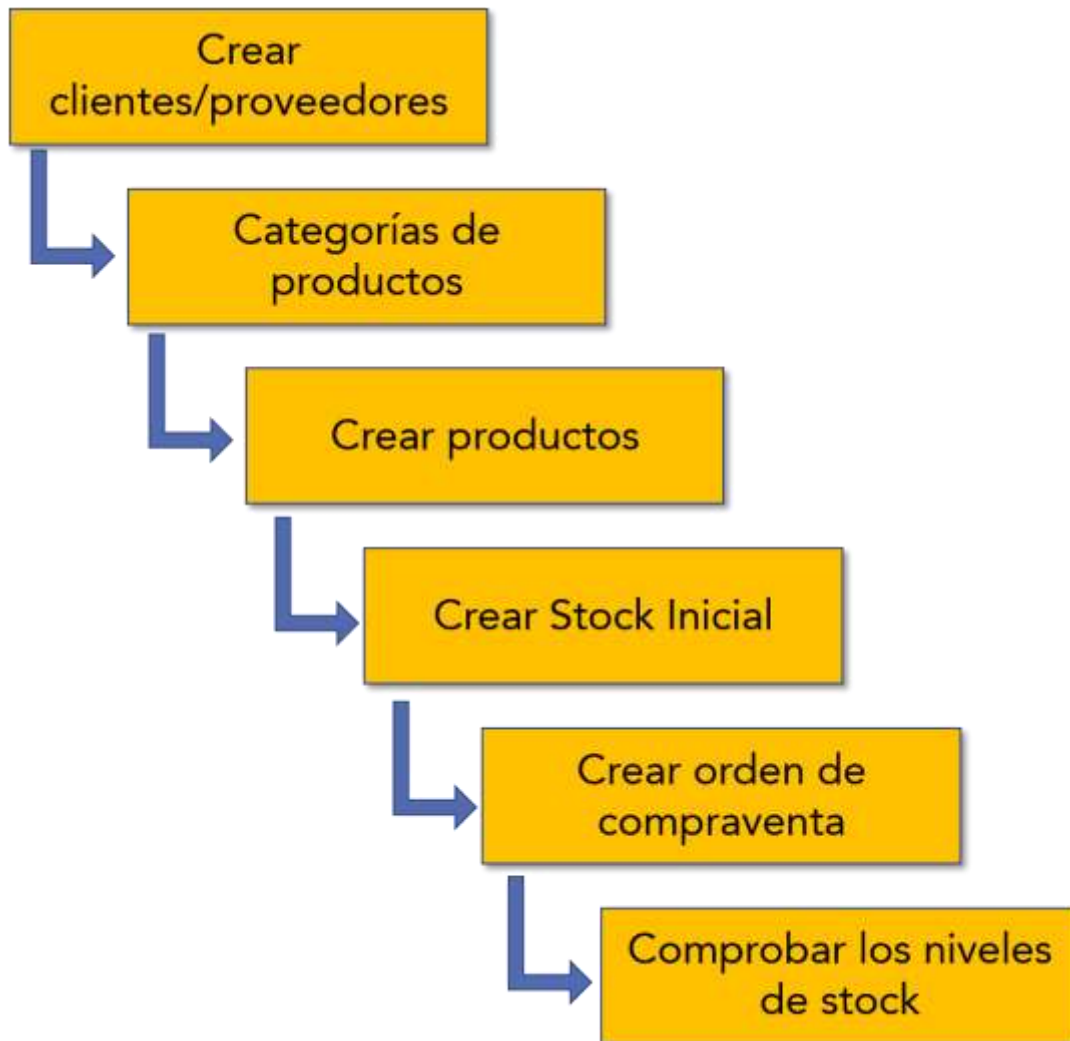
- **Informes**

En **PgAdmin III** contamos con la opción de poder mostrar bastantes informes a cualquier nivel de definición. La forma de poder acceder a estos informes es a través del menú contextual, seleccionando en el elemento **Reportes**.

## 1.2. Interfaces de entrada de datos y de procesos.

### Gestión de compra- venta

- Plan de formación



- Problema planteado

En este apartado, tenemos que saber cuál es el problema que se nos presenta. En este caso, nos encontramos con una empresa dedicada a vender distintos productos informáticos a los clientes mientras se abastece de distintos proveedores. Lo que se pretende es realizar la compra- venta de productos con las correspondientes facturas para los clientes, llevar el control del stock disponible en cada momento e ir lanzando de forma periódica los informes de ventas y compras realizadas, además de ir comprobando el estado del almacén.

- **Creación de socios**

Lo más recomendable a la hora de crear a los distintos socios es crear un conjunto de categorías en las que podamos colocar a cada uno. En un primer momento, las principales categorías van a ser, clientes y proveedores y, una vez que ya se hayan creado, podremos incluir a los socios.

**Primero vamos a comenzar por los proveedores ya que conocemos todos sus datos. Después, iremos añadiendo a los distintos clientes** en el momento en el que realicen su primera venta. Cuando lo registremos la primera vez, no necesitaremos volver a repetirlo.

- 1) Creamos las categorías en **Ventas/Configuración/Libreta de direcciones/Categorías de empresa**. Como mínimo, debemos crear una para cliente y otra para proveedores.
- 2) A continuación, añadiremos los proveedores a los que realizamos las compras mediante **Ventas/Ventas/Libreta de direcciones/Clientes**.

De esta forma, podemos añadir tanto a proveedores como a los compradores seleccionando el botón de **Nuevo**.

No debemos olvidar seleccionar, en la ventana de selección, la opción de **proveedor** para que, de esta forma, podamos realizar las búsquedas que deseemos.

- 3) Por último, añadimos los clientes a nuestra base de datos. Al igual que en el apartado anterior, no podemos olvidar en la ventana de selección la opción de **Cliente**.

- **Creación de las distintas categorías**

En el caso práctico que estamos desarrollando de una tienda informática, es fundamental tener organizados de forma precisa los diferentes productos, dónde los tenemos y cómo los podemos clasificar. De tal forma que, lo primero que vamos a llevar a cabo va a ser, una vez introducidos los clientes y proveedores, crear una serie de categorías para realizar esta clasificación de todos nuestros productos.



Realizaremos esta clasificación atendiendo a los estándares del mercado por lo que lo más recomendable será utilizar una estructura jerárquica con un padre único: **Todos los productos**. Esta categoría ya está definida por defecto, por lo que tendremos que cambiarle el nombre (**All products**).

Llevaremos a cabo esta gestión, siguiendo los pasos **Ventas/Configuración/Productos/Categorías de productos**

En cuanto al control de stock, lo realizará el programa de forma automática al realizar las compras o ventas. Por lo que sólo tendremos que realizar esta tarea la primera vez y cuando modifiquemos productos del catálogo. Gestionaremos los diferentes productos a través de **Ventas/Productos/Productos (Nuevo)**

A la hora de crear un determinado producto, lo primero que debemos será almacenar su nombre, seguido de una referencia. Podemos establecer también si el producto puede ser: comprado, vendido o las dos cosas.

Seguidamente, estableceremos el tipo de producto que deseamos crear donde, **Consumible**: se refiere a que no se gestiona el stock, por lo que la cantidad del mismo es ilimitada.

- **Almacenable**: sí gestionará el stock del producto.
- **Servicio**: lo utilizaremos cuando no se refiere a un producto lo que hemos dado de alta.
- A continuación, podemos determinar el precio correspondiente a lo que podemos comprar (**Precio coste**), el precio a los que lo vamos a vender (**Precio venta**).
- Si necesitamos algún mecanismo que calcule el precio final de venta (**Precio estándar**) y si se tenemos que recalcular con abastecimiento (**Precio medio**).
- Para finalizar, en la pestaña de Proveedores, podemos incluir la información necesaria con sus correspondientes existencias mediante el botón de **Actualizar** que se encuentra en el apartado de Stocks.

### • Órdenes de compras

En este apartado nos vamos a encargar de la compra- venta de los diferentes productos, basándonos en lo que le compramos a los proveedores para después, venderlo a los diferentes clientes. Por lo tanto, es fundamental que, para poder vender, tengamos existencias de los distintos productos, por lo que, lo primero que vamos a realizar va a ser el aprovisionamiento de los mismos, que lo podemos determinar desde la ventana de **Compras**.

A continuación, desde **Compras/Compras/Pedidos de compras** podemos crear la orden de compra en la que estableceremos el proveedor junto con los productos que solicitemos.

Una vez finalizado el albarán, podemos generar los costes totales mediante el botón de **Calcular**. Por lo que ya podemos validar el pedido, utilizando el botón **Convertir a pedido de compra** y generar la factura y el albarán de entrada correspondiente.

Cuando tengamos disponible la mercancía, continuaremos de la siguiente forma:

- Mediante la pestaña Almacén/Gestión de almacenes/Albaranes de entrada localizaremos el alabrán correspondiente de los productos.
- Para mostrarlo, basta con hacer doble clic sobre el alabrán y seleccionar la opción de Procesar.
- A continuación, podemos validar la llegada (Validate) y así nos aseguraremos de que todo es correcto. Las cantidades se

### • Órdenes de ventas

Para crear un nuevo pedido de venta, lo haremos a partir del menú **Ventas/VentasPedidos de ventas (Nuevo)**. Es muy parecido al pedido de compras, por lo que debemos establecer los datos correspondientes del cliente que realice el pedido y añadir los productos que éste desee a través del botón de **Nuevo**.

En cada línea de vente, iremos añadiendo el producto, la cantidad, si tiene algún tipo de descuento, etc. Además, también podremos especificar si es un pedido o si lo cogemos directamente del stock (**Método de abastecimiento**).

En caso de ser un pedido, no es necesario que gestionemos el pago de la factura hasta que se recoja el correspondiente albarán de entrada después de la compra y el cliente lo recoja.

Si lo vendemos desde el stock, el sistema debe avisar si no quedan unidades disponibles en el almacén. En caso de tener disponible, podemos enviar el producto y emitir su factura.

Una vez que el pedido esté completo, calcularemos el precio mediante el botón **Calcular** y lo validamos **Confirmar pedido**.

Se debe generar el albarán de salida junto con la factura cuando seleccionemos **Crear la factura final**.

- **Gestión de devoluciones**

Es bastante común, tanto cuando nosotros compramos como cuando un cliente nos compra, que se produzcan devoluciones. Para realizar este proceso de forma correcta, es conveniente que diferenciamos entre dos tipos de casos:

- 1) Cuando una factura no está pagada.
- 2) Cuando la factura ya ha sido abonada.

En el primer caso, si aún no hemos gestionado el pago y se encuentra en estado de borrados, sólo tenemos que eliminar la factura, cancelando de esta forma el albarán y el pedido.

Si, por el contrario, ya hemos realizado la gestión de pago, no podemos cancelar la factura, sino que tendremos que rectificarla. Este proceso de rectificación, lo vamos a llevar a cabo, buscando la factura antigua (**Contabilidad/Clientes/Facturas del cliente**), abrirla y crear una nueva para rectificar la anterior. Por lo que debe tener la devolución del importe, y volver a pasar por la aprobación, pago y validación de una factura normal.

- **Informes de compras y ventas**

Es muy importante que realicemos los diferentes informes de las compras y ventas realizadas para tener todo documentado. En OpenERP, disponemos de una serie de herramientas que nos realizan este tipo de trabajo, en la opción de **Compras** o **Ventas** en el menú **Informes**.

- Para ventas, se crea el informe **Análisis ventas**
- Para las compras, disponemos de dos informes, **Análisis compras** y **Análisis recepciones**

En los dos casos contamos con la opción de poder ver una lista con los diferentes casos que podemos filtrar según el proceso seleccionado. También podemos realizar una vista gráfica del análisis realizado mediante **Vista gráfica**.

## Gestión del almacén

### • Introducción

La gestión de almacén o stock de productos está basada en dos conceptos fundamentales relacionados entre sí, como son: la Ubicación y el Almacén.

Podemos definir un almacén como una localización física de elementos de stock, pudiendo dividirse cada uno de ellos en diferentes ubicaciones (secciones). De tal forma que, un almacén puede estar formado, como mínimo, por una ubicación.

Los distintos elementos se pueden ir desplazando entre las diferentes ubicaciones, dejando siempre constancia de ello en el sistema, a través de los albaranes. Podemos diferenciar tres tipos de albaranes:

- **De entrada:** utilizado para la recepción de mercancías a un almacén.
- **De salida:** utilizado para la salida de productos de un almacén.
- **Externos:** utilizado para moverlos entre los distintos almacenes o entre las propias ubicaciones de un almacén.

Para la realización de las distintas configuraciones existentes de almacenes, diferenciamos tres ubicaciones posibles:

- **Ubicaciones físicas:** hacen referencia a la estructura física de un almacén.
- **Ubicaciones de socios:** se encargan de ajustar los stocks vendidos y los que se compran.
- **Ubicaciones virtuales:** las que se utilizan mientras se produce el desplazamiento de materiales entre las diferentes fases.

### ○ Ubicación

Corresponde a una parte de la estructura jerárquica que permite representar la sección de un almacén. Podemos diferenciar los siguientes tipos de ubicación a la hora de crearnos una estructura:

- **Vista:** utilizada sólo para organizar, nunca para contener stock. Esta ubicación puede contener otras ubicaciones.
- **Clientes y proveedores:** se refiere a ubicaciones virtuales que permiten representar la salida de los productos y la entrada de material.
- **Interno:** permite gestionar el stock.

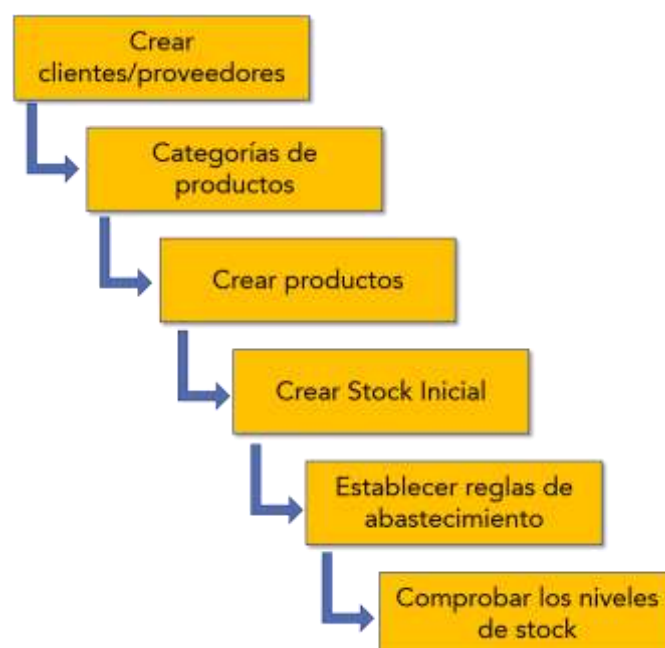
- **Inventario:** utilizado para gestionar las distintas correcciones de forma manual del inventario.
- **Producción:** utilizado para gestionar el material básico de un producto junto con la producción del mismo.
- **Procurement** (Abastecimiento): ubicación temporal que podemos utilizar para realizar la compraventa de algún material.
- **Tránsito:** utilizada en compañías multinacionales mediante movimientos de las diferentes sedes.

Podemos fijar los datos de las ubicaciones: procurement, inventario y producción en la ficha del producto. Y los datos de recepción y reparto en la de socios. La configuración para cada tienda, mediante **Ventas/Configuración/Ventas/Tiendas**.

#### ○ Almacén

Utilizado para agrupar distintas ubicaciones pudiéndolas gestionar conjuntamente. De tal forma que, un almacén consta de una ubicación de entrada (de donde vienen las mercancías), una de salida (hacia dónde van los productos) y una ubicación de stock desde donde se realiza la venta de productos.

#### ○ Plan de formación



### ○ Problema planteado

Seguimos con el mismo problema que planteábamos en el primer apartado, de una tienda de informática, supongamos ahora que se desea abrir una nueva tienda en otro lugar.

Para conseguir mejorar la gestión se van a crear dos almacenes físicos, uno en cada tienda y otro global para que sea desde donde se distribuya a ambas tiendas. Se pueden almacenar los diferentes productos en el global y, desde ahí, abastecer a ambas tiendas de los dos sitios diferentes

### ○ Creación de la estructura del almacén

1. Lo primero que vamos a hacer es crearnos las ubicaciones necesarias, ya que las tres van a depender de la ubicación padre (OpenERP). Nos crearemos una para cada tienda y otra para el stock general, todas van a ser ubicaciones internas.
2. Ahora podemos definir la estructura real que tiene el almacenaje con tres almacenes diferentes: uno para cada tienda y, el general.
3. Cada almacén debe tener: las ubicaciones de entrada y salida, y el stock.
4. Podemos modelar las distintas tiendas físicas incluyendo en cada uno el almacén que puedan utilizar.
5. Llegados a este punto, podemos añadir las categorías junto con los productos necesarios. No podemos olvidar establecer el proveedor.
6. Podemos crear el stock que vamos a tener inicialmente, utilizando la ficha del producto o, a la hora de realizar alguna compra.
7. Finalmente, podemos fijar las reglas de abastecimiento automático utilizadas, abriendo un determinado producto y seleccionando la opción de **Reglas de stock mínimo**.

### ○ Uso del almacén

Una vez que ya hemos creado la estructura y el almacén se encuentra abastecido, podemos pasar a realizar las ventas desde las diferentes tiendas.

Primero, vamos a vender en una tienda y, al detectar que no se encuentran stocks del producto solicitado, tendremos que traspasar productos desde el almacén general.

Cuando finalicemos el pedido de venta y comprobemos que no tiene stock disponible, debemos hacer una petición de material utilizando un albarán interno. En este albarán, debemos indicar las ubicaciones, tanto origen como destino, además de la cantidad que sea necesaria. Trataremos este albarán como uno más, por lo que lo debemos procesar y validar.

Una vez que finalicemos el albarán, los datos habrán pasado ya desde el almacén general hasta la tienda que los requería. Por tanto, podemos finalizar el albarán de salida que creamos con la compra junto con su factura correspondiente.

Mediante la opción **Almacén/Planificación/ Excepciones abastecimiento**, podremos comprobar la excepción que ha provocado esta situación.

#### ○ Informes del almacén

Es fundamental realizar un buen seguimiento del stock para poder garantizar el funcionamiento correcto de una determinada empresa y, de esta forma, aumentar la satisfacción por parte del usuario.

Para ello, debemos realizar diariamente las comprobaciones del estado de los albaranes, excepciones de abastecimiento, peticiones de compra, etc que podemos llevar a cabo mediante las herramientas para gestión y análisis.

A través de la opción de **Almacén/Informe/Tablero/Tablero de almacén** podemos encontrar el tablero principal del almacén.

Para comprobar las cantidades de las distintas ubicaciones, diferenciamos:

- **Análisis de movimiento**, para comprobar todos los trasposos de material que se han producido en los diferentes almacenes.
- **Análisis de inventario**, podemos comprobar cómo se han repartido los productos por las distintas ubicaciones.

Y, la última herramienta que vamos a definir **análisis detallado de movimiento**, a la que podemos acceder mediante **Almacén/ Trazabilidad/Movimientos de stock**, nos permite la opción de poder comprobar todos los cambios de stock que se hayan producido en los diferentes almacenes e ubicaciones.

### 1.3. Cálculos de pedidos, albaranes, facturas, asientos predefinidos, trazabilidad, producción. 3.4 Gestión de contabilidad

#### ○ Introducción

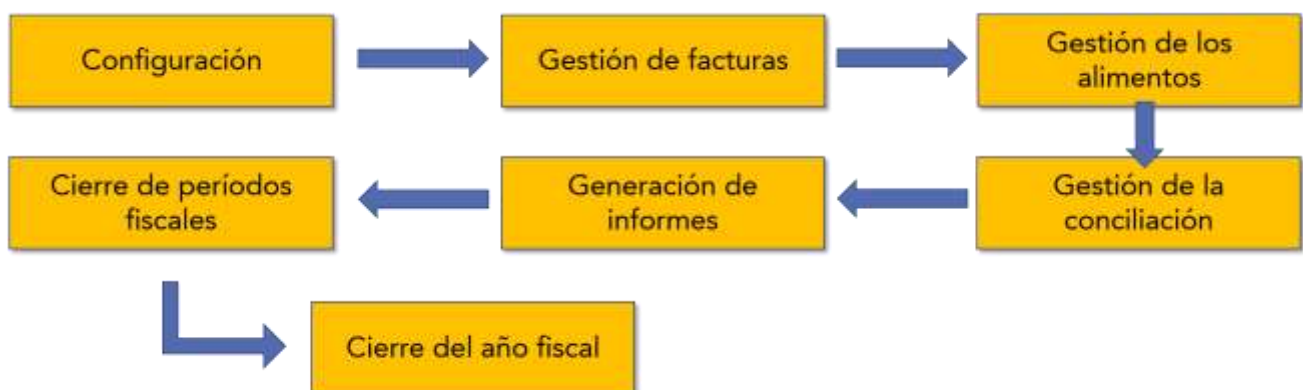
La contabilidad se encuentra completamente integrada con los sistemas de compras y ventas, permitiendo el transcurso de la información en tiempo real.

La contabilidad, está basada en dos fuentes principales para conseguir así, un correcto funcionamiento:

- La configuración de los distintos ejercicios y periodos, diarios, plan contable, impuestos, plazos y tipos de pago.
- Actividad diaria del sistema que va a generar la información que se genera de forma automática y la introducida de forma manual por el correspondiente contable.

La unión de todos ellos, van a producir la correcta gestión de contabilidad empresarial, permitiendo conciliar facturas, visualizar balances, etc.

#### • Plan de formación



#### • Problema planteado

En este apartado de la gestión de contabilidad, no vamos a poner ningún nuevo supuesto que tengamos que implementar. Seguiremos con la tienda informática de apartados anteriores, aprendiendo los distintos procedimientos básicos que puede utilizar un contable, familiarizándonos con ellos y aprendiendo su forma de trabajar.



Aunque nosotros no nos vamos a centrar en realizar tareas contables, sí que debemos ser capaces de poder resolver las dudas referentes a su utilización que puedan surgir, por tanto, debemos ser conocedores del tema en cuestión.

- **Configuración del sistema**

### 1. Ejercicios fiscales

Lo primero que debemos crear es el ejercicio fiscal con el que vamos a trabajar que suele ser siempre, un año natural. Aunque, cada empresa tiene libertad de comenzar y terminar este ejercicio en el momento que desee.

Lo que sí es común a todos los ejercicios es que deben estar formados por periodos que van a coincidir con aquellos momentos de liquidación de la empresa (generalmente, por trimestres para PYMES).

Para crear el año fiscal, podemos hacerlo mediante **Contabilidad/ Configuración/ Contabilidad financiera/Periodos/Ejercicios fiscales**, y nos debe aparecer un listado que contenga todos los ejercicios fiscales que se han definido junto con su estado, que puede ser abierto o cerrado. A través del botón **Nuevo**, podremos crear uno nuevo.

Introducimos una serie de valores obligatorios, como pueden ser el nombre del ejercicio, código, y las fechas de inicio y fin. También podemos indicar si deseamos crear los periodos de forma automática o no.

### 2. Periodos

Forma parte del ejercicio fiscal y es posible que coincida con la liquidación de impuestos realizada por la compañía que determine la ley actual, generalmente suelen ser trimestrales o mensuales.

Los periodos definen un principio y un fin, y no se pueden solapar por defecto. Si lo deseamos, podemos eliminar esta condición indicándolo en el momento de su creación, cuando utilicemos sólo periodos de cierre y apertura del año.

Si lo que pretendemos es crear los diferentes periodos de forma automática, lo podemos hacer mediante, **Contabilidad / Configuración / Contabilidad financiera / Periodos / Periodos** y pulsamos el botón de **Nuevo**.

### 3. Diarios

Podemos definir un diario como un libro contable en el que podemos registrar los asientos que se vayan produciendo durante el funcionamiento diario de una empresa.

Lo más lógico es tener un libro asociado a una empresa aunque OpenERP ofrece muchas más posibilidades y permite crear más de un libro para una misma empresa.

La opción más recomendable de la documentación es crear un diario distinto para cada operación y, como mínimo, tres (uno para compras, otro para ventas y otro para efectivo). De esta forma, aparecen los siguientes tipos de diarios:

- **General.** Nóminas, liquidaciones de impuestos, movimientos de capital, etc.
- **Ventas.** Registros de ventas y facturas de los clientes.
- **Compras.** Registros de compras y facturas del proveedor.
- **En efectivo o de caja.** Los que gestionan el dinero.
- **Abono de.** Para gestionar las devoluciones de compras y ventas.
- **Banco y cheques.** Gestiona las cuentas bancarias y utilización de cheques.

A la hora de realizar las diferentes gestiones, podemos realizarlo mediante la opción de **Contabilidad / Configuración / Contabilidad financiera / Diarios / Diarios**

Rellenando como campos obligatorios el nombre y código. Además, es conveniente que indiquemos el tipo de diario y sus cuentas contables por defecto que son las que se van a utilizar a la hora de crear los diferentes asientos, un modo de visualización para gestionar los asientos junto con el mecanismo de numeración de los mismos.

A través de la pestaña **Controles de asiento**, podemos acceder a un determinado diario, incorporando un mayor nivel de seguridad.

### 4. Impuestos

Lo normal cuando instalamos un programa es que se configuren los impuestos necesarios de forma correcta, aunque algunas veces, vamos a tener que ir modificándolos si la legislación de modifica. Podremos acceder mediante **Contabilidad / Configuración / Contabilidad financiera / Impuestos / Impuestos**

Los datos obligatorios van a ser el nombre, cuándo se va a aplicar, el tipo de impuesto que es y el importe que se le va a aplicar.

## 5. Mecanismos de pago

Dentro de los mecanismos de pago podemos hacer referencia a las distintas configuraciones que podemos realizarle a los mismos, a través del menú **Contabilidad/Configuración/Varios**

- En el primer apartado, **Plazos de pago** que pueden determinar el tiempo que tarda una factura en ser abonada.
- En el segundo, podremos configurar los diferentes tipos de pago que pueden utilizar las facturas.
- El último apartado hace referencia a los modos de pago que van a permitir conformar las órdenes de pago como pueden ser las remesas.

Y para concluir, destacaremos también los tipos de **Facturación** que pueden existir para un determinado cliente.

## 6. Plan contable

Se refiere a la estructura jerárquica de las distintas cuentas contables. Para crear un plan contable, debemos saber que éste no se crea sólo, sino que, podemos hacerlo mediante la utilización de las plantillas existentes para OpenERP.

Una vez creado el plan, es raro que lo modifiquemos, ya que prácticamente, lo diseñamos a nuestro gusto. A no ser que cambien las necesidades de la empresa.

Gracias a la opción de **Configuración / Planes contables / Plan contable**, vamos a poder acceder a todos los asientos ya ordenados según las cuentas de la gestión del plan contable.

## 7. Cuentas

Una vez que ya hemos creado el plan contable podemos encontrar alguna situación en la que debemos modificar alguna cuenta porque hay ocurrido algún cambio.

En este caso, a través de **Contabilidad / Configuración / Contabilidad financiera / Cuentas / Cuentas** y podremos visualizar el listado de cuentas con la opción de filtrar según alguna condición. Si deseamos crearnos una nueva cuenta, seleccionaremos la opción de **Nuevo**.

- **Gestión de facturas**

Mediante las facturas, nos podremos hacer cargo de justificar los pagos. Es fundamental que cada gasto esté justificado y podemos realizarlo de forma automática, mediante el propio sistema informático.

Podemos diferenciar la gestión de las facturas de los clientes de las de los proveedores.

- **Gestión de factura de los clientes**

La seleccionamos mediante **Contabilidad/Clientes**

Y deben aparecer aquellos elementos necesarios para su gestión. Primero, en **Facturas de cliente** mostrará un listado con todas las facturas que están dadas de alta en el sistema, tanto si están pagadas como si no, además de las que estén todavía pendientes de validar. De tal forma que podemos acceder a cualquiera de ellas y realizar las acciones oportunas.

- **Gestión de factura de los proveedores**

Es un proceso bastante similar al de los clientes con la salvedad de que es bastante frecuente incorporar facturas externas al sistema. Dentro de estas facturas van a aparecer aquellas que esten dadas de alta, tanto las que se hayan realizado de forma manual como las automáticas.

En el caso en el que deseemos añadir otra nueva factura, debemos incluir en el diario la factura, el proveedor, la referencia de factura junto con el importe total. Los demás campos se rellenarán de forma automática.

- **Gestión de asientos**

Podemos definir un asiento como un conjunto de apuntes que permiten representar algún movimiento contable. OpenERP, ofrece la posibilidad de crear los distintos asientos, tanto de forma manual, como automática. Aunque es más frecuente de esta segunda forma, ya que se puede crear en el momento en el que se realice el pago correspondiente.

En caso de utilizar la forma manual, podemos hacerlo a través de **Contabilidad / Asientos contables** y, así podemos acceder a la gestión de apuntes y asientos.

Los dos tipos de operaciones más importantes que podemos utilizar con los asientos van a ser:

- **Asentar.** Validar varios asientos de un determinado diario y periodo de tiempo.
- **Conciliación de asientos.**

Para los dos casos, seleccionaremos los elementos necesarios en los apuntes contables que podemos encontrar en el menú lateral derecho y una vez ahí, ya podremos llevar a cabo la operación. En este menú, también disponemos de la opción de poder romper la conciliación de distintos asientos ya conciliados, permitiendo de esta forma la modificación de los mismos.

#### ○ **Generación de informes contables**

Como ya hemos indicado en apartados anteriores, la herramienta más importante en la gestión contable es el tablero de contabilidad. Podemos acceder a él mediante **Contabilidad / Informe / Tablero / Tablero de contabilidad**

En el tablero podemos apreciar en su parte superior aquellas facturas que todavía no han sido aprobadas y, como ya sabemos, una de las tareas más importantes en la contabilidad es que, una factura debe ser aprobada antes de ser abonada, por lo que es conveniente que se apruebe cuanto antes.

La contabilidad del sistema presenta una gran cantidad de informes que pueden generarse de forma automática. Algunos de los más conocidos pueden ser, entre otros:

- **Informes contables**, entre los que destacamos el libro mayor, balance de sumas y saldos, ganancias y pérdidas y la hoja de balance. Generan un fichero PDF que se puede imprimir.
- **Diarios**, que podemos seleccionar si lo vamos a crear de forma individualizada por diarios, sólo los diarios generales o un diario centralizado
- Modelos de impuestos para la Hacienda Española: 340, 37, 349. Una vez introducidos los datos necesarios, se realizan de forma automática.

### ○ Cierre contable

Cuando finaliza un ejercicio fiscal, el contable correspondiente debe realizar una serie de procedimientos especiales. Es bastante frecuente que, con el fin del periodo fiscal se calculen una serie de informes necesarios para que se realice el pago de impuestos.

Sí que es cierto que, cerrar el año fiscal es un proceso más complejo. Como hemos indicado, el contable es el encargado de realizar las diferentes tareas sobre el sistema de forma correcta, comprobando que no falte nada por contabilizar y que no existan apuntes incorrectos, entre otras cosas. Una vez realizadas las distintas comprobaciones ya puede pasar a crear un inventario físico de existencias. En este apartado, se debe amortizar los activos que tengamos, ajustar las diferentes cuentas que puedan influir sobre el ejercicio y calcular el impuesto de sociedades. Por último, debe reenumerar los asientos contables para que sean consecutivos.

A continuación, puede pasar a cerrar el ejercicio, siguiendo una serie de pasos, como pueden ser, entre otros:

- Una vez finalizadas las tareas, cerraremos el ejercicio posicionándonos en **Contabilidad / Configuración / Contabilidad financiera / Periodos / Ejercicios fiscales** comprobando que los asientos de apertura y cierre estén creados. Si no lo están, lo realizaremos mediante **Create Open/Close and PyG Periods**
- Seguiremos con el cierre del ejercicio desde **Contabilidad / Procesamiento periódico / Fin de periodo/ Close Fiscal Year**, seleccionando los años de apertura y cierre. Debemos asignarle un nombre que sea único.
- Podemos configurar las diferentes acciones que deseemos hacer que no tienen porqué realizarse todas a la vez, y continuamos con el asistente.  
Entre las diferentes opciones podemos encontrar la de realizar los movimientos de pérdidas, ganancias y pérdidas netas, cierre del año fiscal actual y apertura del siguiente año.
- Una vez realizada las operaciones, debemos **Guardar** el cierre y no editarlo más.
- Es fundamental que antes de confirmar el cierre, validemos cada movimiento que ha creado el asistente.

- Y para finalizar, confirmamos el cierre seleccionando el botón de **Conrfim**.
- Si tenemos que realizar alguna modificación de algún asiento, tendremos que:  
Primero, reabrir el año correspondiente. A continuación, modificar los datos y, por último, volver a cerrar el ejercicio. Para ello, seleccionaremos **Contabilidad/Procesamiento periódico/Fin de periodo/Fiscal Year Closings**, y editamos el ejercicio en cuestión.

## 1.4. Gestión de los recursos humanos

- Plan de formación



- Problema planteado

La gestión de Recursos Humanos (RRHH) va a tener entre sus principales funciones:

- Crear
- Modificar
- Mantener

Tanto a los empleados como sus contratos. Además del control de asistencia al trabajo, organización en caso de ausencias, gestión de nóminas y obtención de informes entre otras funciones.

Vamos a suponer un caso práctico a modo de ejemplo en el que TeclitaSoft va creciendo y necesita contratar personal de contabilidad para la Gestión Contable y de recursos humanos, externos a la empresa. Tendremos que dividir en dos partes, dentro de la gestión de RRHH, las cinco personas que trabajan en la empresa. Por un lado, jefe y empleados y, por otro, los departamentos de Ventas, Técnicos y Dirección.

La dirección sólo contará con una persona, mientras que Ventas y Técnicos van a constar de dos personas cada uno.

- Los contables van a ser los encargados de diseñar la estructura, los contratos de los empleados, nóminas necesarias e informes.
- Los emplados deben justificar su asistencia al trabajo fichando en la página correspondiente del sistema, solicitar vacaciones, informar ante cualquier ausencia, etc.

### • Configuración del sistema

Antes de comenzar a buscar una solución ante el problema planteado, realizaremos la configuración del sistema, añadiendo los datos correspondientes junto con la estructura de categorías y puestos ofrecidos.

En principio, configuraremos los datos de asistencia para poder, a continuación, diseñar la estructura de la misma.

1. Mediante la opción de **Recursos Humanos / Configuración / Asistencia** podemos establecer las causas que se van a permitir en la empresa.

En un primer momento podremos añadir el nombre de la acción determinada y si va a ser de entrada o de salida.

2. En el menú **Recursos Humanos / Configuración / Vacaciones / Tipos de ausencia** podemos seleccionar dos tipos de ausencias por las que pueden faltar al trabajo. Cuando un empleado falta a su puesto de trabajo, el sistema interpreta que puede estar de vacaciones, aunque puede ser por algún otro motivo.

3. En este apartado ya podemos definir la estructura de la empresa por categorías, mediante,

**RecursosHumanos / Configuración / RecursosHumanos / Empleados / Categorías**



Es conveniente que, para cada categoría, indiquemos el nombre junto con el padre al que corresponde y asegurarnos de que, cada empleado, debe pertenecer a una de estas categorías como mínimo. En el momento en el que necesitemos visualizar la estructura, lo haremos a través de, **Estructura de categorías**

4. El apartado de **Recursos Humanos / Configuración / Recursos Humanos / Empleados / Marital Status**. El estado civil de los empleados viene definido por defecto y, en caso de necesitarlo, se puede modificar sin problema.

5. **Recursos Humanos / Configuración / Recursos Humanos / Departamentos**

Nos permite realizar cualquier tipo de modificación sobre la estructura departamental que determina la empresa. Podemos modificarlo hasta que nos acerquemos a la estructura actual de la empresa pensando en un futuro próximo.

6. **Recursos Humanos / Configuración / Recursos Humanos / Contratos**

Nos permite realizar las modificaciones oportunas sobre el tipo de contrato de los empleados. Podemos diferenciar entre tres submenús:

- Uno para establecer el tipo de contrato.
  - Otro que corresponde al tipo de salario.
  - Y el último, que determina el tiempo de pago.
- **Gestión de empleados y contratos**

Una vez configurados todos los aspectos principales para la gestión de empleados y contratos, podemos crearlos y relacionarlos.

Lo primero que debemos hacer es crear un usuario del sistema, mediante **Administración/Usuarios** Estos usuarios deben tener, al menos, un nombre y una clave.

A continuación, podemos crear los distintos empleados en **Recursos Humanos/Recursos Humanos/Empleados** seleccionando la opción de **Nuevo** Rellenamos los datos más importantes de los empleados.

Por último, para crear un contrato, es fundamental indicar el nombre del empleado al que se le hace el contrato, el salario, tipo de salario y tipo de contrato. Además, también debemos señalar la fecha de inicio y grabar el contrato.

Y para finalizar, debemos asignar los directores de cada departamento ya que todos los empleados ya están dados de alta en el sistema. Lo haremos en el menú de **Recursos Humanos / Configuración / Recursos Humanos / Departamentos**

- **Gestión de asistencia**

Como hemos indicado en anteriores apartados, un usuario puede asistir al trabajo o faltar de una forma justificada. En cualquiera de los casos, **debemos gestionar la asistencia por dos vías distintas:**

1. De forma monetario
2. Organizativa

Las dos formas deben contar con un **documento** que las justifique.

**El mecanismo más utilizado por las empresas es la justificación mediante fichaje de los empleados, de esta forma, el empleado es el responsable de su asistencia o de su ausencia en su puesto de trabajo.**

De la misma forma, el empleado, en caso de ausencia al trabajo, puede ser por dos motivos diferentes: vacaciones o falta justificada. Las vacaciones son un motivo claro y regulado pero la falta justificada, engloba a su vez, numerosas posibilidades, como pueden ser, entre otras, asistencia médica, incapacidad temporal, reuniones, visitas a clientes, etc.

**Para OpenERP, cualquier falta, cuenta como vacaciones.**

- Para registrar la entrada, tendremos que hacerlo mediante **Recursos Humanos/Servicios/Registrar entrada/salida** y utilizaremos la opción de **Registrar entrada** para introducir el usuario y la clave.
- Y, en el momento de la salida, el usuario debe realizar el mismo proceso mediante la opción **Registrar salida**.
- Mediante la opción de **Recursos Humanos/Seguimiento de tiempo** podemos asignar una serie de permisos a los diferentes empleados para conseguir, de esta forma que, éstos tengan una visión más amplia de la estancia y podrán entrar en el menú **Mi hoja de asistencia** para visualizar el listado semanal de aquellos servicios que se han realizado.

La jornada laboral correspondiente en la ficha de cada empleado, va a determinar la cantidad de días libres que le corresponden por semana asignados y validados. Para las demás faltas de trabajo que pueda tener (vacaciones o justificadas), debe solicitarlas al sistema para ser aprobadas por algún responsable de la empresa.

- Cuando es una **falta justificada**, crearemos una **petición de ausencia**
- Si se solicitan **vacaciones**, pediremos una **petición de asignación**

El sistema es el responsable de cerrar las correspondientes hojas de trabajo y validarlas. De esta forma se asegurará que las nóminas se realicen de forma correcta.

Mediante **Confirm** podremos confirmar la hoja de trabajo y la persona que sea responsable es el encargado de validarla. Para tal fin, debe abrirla y seleccionar la opción de **Aprobar**.

#### ○ Nóminas

Para la realización de las nóminas, existen dos tipos de mecanismos: general y específico. En esta Unidad Formativa, estudiaremos el general ya que es más sencillo.

Para ello, debemos tener configurados los datos del contrato para poder crear las diferentes nóminas. Lo primero que tenemos que hacer es cerrar las hojas de trabajo y, a continuación, seleccionaremos el menú **Recursos Humanos / Nómina / Nómina de los empleados** para crearnos las diferentes nóminas de los empleados.

Esta opción nos puede servir en el caso de tener pocos empleados en la empresa, pero, si es una empresa mayor, con más empleados, es más conveniente generar las nóminas para los diferentes conjuntos de empleados, mediante la opción de **Recursos Humanos / Nóminas y anticipos**.

Para crear una nueva nómina, seleccionamos el botón de **Nuevo** y, a continuación, introducimos los datos del empleado, el diario contable (**Diario de gastos**), diario desde el que vamos a hacer el pago (**Diario Bancario**) y la fecha correspondiente.

En la opción de **Comprobar hoja** podemos realizar una vista previa de la nómina y, si todo es correcto, podemos **Aprobarla**. De esta forma, la nómina pasa a ser válida y ya está creada, aunque no se ha pagado. El pago es otro apartado posterior en el que se crearán los asientos contables y se envía la orden al banco para que éste se realice.

- **Informes**

Podemos realizar numerosos informes para controlar la gestión del personal. Como ya hemos indicado, una de las herramientas más importantes para este fin es mediante el tablero o los propios informes. Ambos los encontraremos en **Recursos Humanos/Informe/Tablero**, siendo algunos de ellos:

- **Horario del empleado** nos va a facilitar un informe con los horarios de un empleado.
- **Horarios de empleados** nos va a facilitar un informe con los horarios de los empleados.
- **Recursos humanos/Informe/Ausencias** proporciona un informe con las ausencias y nóminas de los empleados.

## 1.5. Herramienta de Monitorización. Incidencias: resolución e identificación.

Todos los sistemas informáticos necesitan tener un mantenimiento en dos niveles diferentes: interno y externo.

En las tareas de mantenimiento podemos encontrar, entre otras, realizar copias de seguridad, actualizar el software, control del sistema y auditoría del mismo. En este apartado nos basaremos en sistemas *OpenERP- PostgreSQL*.

- **PostgreSQL**

Para las empresas de hoy en día, **la función más importante es su información correspondiente**. Esta información se puede desprender o inutilizar provocando grandes pérdidas. Ante esta situación, es fundamental realizar una buena gestión de la base de datos, estableciendo una correcta política de seguridad, la distribución de datos y control de acceso.

PostgreSQL emplea varios ficheros de configuración, como puede ser **postgresql.conf** que establece los diferentes datos de funcionamiento. Y, dentro, cuenta con una serie de secciones en las que se encuentran las auditorías.

Si nos desplazamos por este fichero, encontramos la sección **ERROR REPORTING AND LOGGING** que se encarga de la configuración de los aspectos de la creación de ficheros para las auditorías.

La sección **When** nos va a indicar cuál es el tipo de mensajes que podemos almacenar en el caso de:

- cliente (client\_min\_messages)
- Para el propio sistema (log\_min\_messages)
- Ante el fallo de una sentencia (log\_min\_duration\_statement)
- Si una sentencia se lleva demasiado tiempo en ejecución (log\_min\_duration\_statment). Se expresa en milisegundos.
- Al final, podemos identificar aquellos elementos de la base de datos que deseemos almacenar para, a continuación, formalizar el formato de la sentencia. Para ello, escribiremos (log\_line\_prefix).

Otra sección bastante importante, es **RUNTIME STATISTIC** que nos ofrece la posibilidad de poder realizar la configuración sobre las estadísticas del uso que ha tenido el sistema y, de esta forma, podremos determinar su rendimiento.

Mediante estas estadísticas, vamos a generar, por una parte, la implicación al sistema de las diferentes consultas (*Query/Index Statistics Collector*) y, por otra, el sistema de auditoría (*Statistics Monitoring*).

#### ○ **OpenERP: auditoría por ficheros**

OpenERP es un paquete compuesto por dos elementos principales como son el servidor y la pasarela Web que lleva a este servidor. Cada una de estas partes tiene su propio fichero de configuración que se encuentran en el directorio **/etc**.

Los parámetros que tenemos disponibles en el **servidor** para poderlos configurar, van a ser:

- **syslog** (=False), si hacemos uso de Linux para una auditoría.
- **logrotate** (=True), cuando el sistema gira los ficheros de log de forma automática.
- **log\_level** (=info), para almacenar una serie de información referente a los distintos eventos de los ficheros.

Para configurar los datos correspondientes en la **pasarela** Web, podemos hacer uso de los siguientes ficheros:

- Ficheros de acceso (log\_access\_file)
- Ficheros de error (log\_error\_file)

#### Nivel de ficheros para acceso y errores.

Esta configuración es bastante más compleja que la anterior, por lo que, podemos consultar más información, desde la página correspondiente.

<http://docs.python.org/library/logging.handlers.html#logging.handlers.TimeRotatingFileHandler>

#### ○ OpenERP: módulo audittrail

El módulo de auditoría que incorpora OpenERP es bastante fuerte, pero a su vez, también resulta un poco complejo ya que puede solicitar información de cualquier parte del sistema. En este apartado, como es muy amplio, nos vamos a centrar de forma más específica en la configuración de este módulo.

Para ello, vamos a tener en cuenta la cantidad de información que tenemos disponible en la correspondiente base de datos, siendo recomendable su utilización durante un periodo determinado de tiempo ya que no es recomendable tener el servidor en producción activo siempre.

La instalación del módulo la realizaremos mediante **Administración/Módulos**. De esta forma, se muestran dos nuevos elementos en el menú **Administración/Informes/Auditoría**.

En el primer elemento, vamos a configurar las diferentes reglas de auditoría, como (*Audit Rules*), pudiendo indicar los objetos que se pueden controlar junto con sus correspondientes acciones. En el segundo elemento, van a aparecer una serie de informes asociados (*Audit Logs*).

Cada regla debe tener un nombre único junto con el objeto que se desea controlar, además de las acciones (lectura, escritura, eliminación, creación, acciones o flujo de trabajo). También podemos establecer los usuarios para los que vamos a llevar el control.

- **OpenERP: control de rendimiento**

Este módulo nos permite determinar los tiempos de ejecución que tiene cada función del sistema.

El sistema nos permite realizar las funciones de instalación de un módulo para recoger los módulos (profiling) para que posteriormente, muestre los correspondientes informes.

Este módulo de control de rendimiento no viene añadido en la distribución estándar, por tanto, tendremos que descargarlo de Internet.

<http://v6apps.openerp.com/addon/5470>

Una vez descargado, lo copiaremos en el directorio **addons** del servidor y, a continuación, actualizar los módulos necesarios.

Este módulo nos va a permitir obtener las estadísticas correspondientes a la ejecución sobre la cantidad de llamadas a los servicios junto con el tiempo empleado para ello, los métodos ORM del servidor, la información asociada a los métodos y servicios ORM y el número de registros devueltos por estos métodos.

Una vez analizada la información, ya podemos realizar los cambios que deseemos en la estructura de nuestro código para, conseguir de esta forma, mejorar el rendimiento.

### ○ Otras herramientas

Podemos finalizar este apartado señalando algunas de las herramientas que también podemos utilizar, como pueden ser:

- <http://docs.python.org/2/library/profile.html>  
Utilizada para crear módulos mediante el profiler de Python, con el que podremos obtener las estadísticas y tiempo de ejecución sobre su uso.
- <http://pgfouine.projects.pgfoundry.org/>  
Analizador de sentencias SQL que se encarga de ejecutar las correspondientes órdenes SQL en la base de datos, mostrando los tiempos y el uso.
- <http://www.postgresql.org/docs/9.2/statics/sql-explain.html>  
Permite mostrar el plan de ejecución para una sentencia determinada.



## 1.6. Procesos de extracción de datos en sistemas ERP- CRM y almacenes de datos.

### ○ Plan de formación



### ○ Problema planteado

En este apartado, debemos proporcionar a los clientes una serie de nuevos servicios, como: el soporte telefónico y la gestión de reclamaciones. Además, debe contar con la posibilidad de poder ofrecer telefónicamente a sus clientes, diferentes ofertas que le puedan interesar.

Todos estos servicios se pueden gestionar desde la plataforma, dejando la información almacenada y comenzando con el proceso de venta si es que se llega con algún cliente a buen fin. De esta forma, podremos dotar a la empresa de un mayor valor.

### ○ Configuración del sistema

Tenemos la opción de dejar las configuraciones que vengán establecidas por defecto o, modificarlas para cada empresa. Utilizaremos la opción de **Ventas/Configuración**

Siguiendo una serie de pasos, como pueden ser:

1. **Iniciativas y Oportunidades/Etapas** nos van a permitir configurar las distintas etapas, el orden, tipo y la probabilidad de que se pueda llevar a cabo.
2. Al crear una iniciativa tenemos que señalar cuál es su principal motivo, mediante la información que reflejan las categorías de iniciativas. A través de **Iniciativas y Oportunidades/Categorías** podemos realizar las diferentes configuraciones.

3. Es importante indicar cuál es el mecanismo de comunicación utilizado entre la empresa y el cliente. **Iniciativas y Oportunidades/Canales**, como pueden ser vía telefónica, sitio web, correo electrónico, entre otros.
4. **Oportunidades/Satges** son las diferentes fases por las que debe pasar un documento.
5. Para buscar nuevos clientes debemos comunicarnos con ellos independientemente de la forma que seleccionemos para ello.
6. En las empresas se van a realizar reuniones a diferentes niveles como por ejemplo, del jefe con los empleados, del comercial con los clientes, etc. Para ello, es conveniente utilizar un calendario interno de gestión y así poder llevar el control de las diferentes reuniones. De esta forma, podemos obtener dos ventajas principales como pueden ser conseguir mayor eficiencia y tener documentado todo este proceso.
7. El servicio de reclamaciones se puede llevar de forma automatizada. Según la ley, debe existir el libro y hojas de reclamaciones.
  - a. **Reclamación/Categorías** atenderemos las distintas categorías y, mediante **Reclamación/Etapas**, definiremos cada una de las etapas que tenga.
8. Para finalizar, vamos a configurar los servicios de ayudas disponibles que podamos ofrecer a los clientes definiendo sus categorías. Mediante **HelpDesk/Categorías** podremos describir los elementos.

#### ○ **Gestión de iniciativas**

El sistema puede diferenciar entre dos tipos de documentos:

##### 1. **Iniciativas**

Podemos entender iniciativa, como una preventa, por lo que puede que se termine concretando o, no. Es decir, la iniciativa debe pasar por un conjunto de fases que pueden terminar con un proceso exitoso o por el contrario, terminar sin llegar a nada.

Sea como fuere, debemos comenzar creando dicha iniciativa mediante **Ventas/Ventas/Iniciativas**, en la que debemos rellenar el campo **Asunto**

de forma obligatoria. De esta forma, vamos a conseguir ordenar los documentos y encontrar el que buscamos.

El documento que se crea puede sufrir modificaciones a lo largo del tiempo según cuál sea su situación en cada momento.

Una vez que el documento ha concluido todas las fases, la iniciativa habrá llegado a su fin, por lo que seleccionaremos el botón de **Cerrado** y podemos dar por finalizada esta tarea.

## 2. Oportunidades

Gracias a las oportunidades podemos ofrecer una estimación sobre las ventas. De esta forma, el primer campo que vamos a rellenar va a ser el **Ingreso estimado**, después del nombre descriptivo **Oportunidad**. Gracias a este valor, vamos a poder determinar las ganancias que no se han realizado al finalizar el año.

El tiempo que dure una oportunidad, vamos a poder mantener contacto con la persona relacionada por distintas vías.

Mediante la opción de **Planificar/Registrar llamada** podemos definir la fecha prevista y determinar si es de entrada o de salida junto con el usuario al que se le realiza la llamada.

En la ventana de **Comunicación e Historial**, la información va a quedar reflejada en un listado y, si seguimos avanzando, llegaremos a **Programar reunión** que, como su propio nombre indica, asignará una reunión en el calendario.

De la misma forma que las iniciativas, la oportunidad también puede tener un final positivo **Marcar ganado** o si no llega a realizar la venta, **Marcar perdido**. En el caso en el que se consiga, pasaremos a **Convertir a presupuesto**.

### ○ Gestión de reclamaciones

Es inevitable que se produzca algún tipo de error en el sistema y no tengamos control para solucionarlo. Por tanto, debemos facilitar un buen mecanismo para las distintas reclamaciones que se puedan producir con idea de que reduzcamos el tiempo en solventar el error. Así, el cliente va a quedar más satisfecho y tendrá una visión positiva del servicio de la empresa.

Para poner en marcha una reclamación, lo haremos mediante **Ventas/Servicio de Postventa/Reclamaciones**

Esta reclamación debe constar de un nombre, fecha de apertura y prioridad que se le asigna. Debemos incorporar toda la información para que se pueda gestionar con el **Responsable**, la etapa en la que se ha producido **Etapas**, los datos de la persona que hace la reclamación, una descripción y a qué objeto se refiere **Referencia**.

Una vez concluida la reclamación, finalizamos y cerramos el proceso.

### ○ Gestión de soporte (HelpDesk)

Es bastante frecuente que las empresas, con el tiempo, vayan ampliando sus oportunidades para ofrecer a los clientes más soporte sobre diferentes productos. Para llevar a cabo esta gestión de soporte, hace falta un tipo de personal cualificado para resolver las diferentes dudas y problemas que se presenten.

Mediante la opción de **Ventas/Servicio de Postventa/Helpdesk y soporte** podemos dar de alta esta opción para poder tratar la petición realizada.

Esta gestión de soporte por el canal establecido, pudiendo cerrar la consulta en cualquier momento, cancelarla o derivarla a otra persona de mayor rango si no lo podemos solucionar (**Escalar**). Mediante la pestaña de **Comunicación e Historial e Información extra**, estableceremos aquellos datos correspondientes a la consulta.

Además, el sistema cuenta con la opción de poder enviar al responsable distintos recordatorios de forma automática.

### ○ Informes

Como hemos indicado en apartados anteriores, los informes son una parte fundamental de los sistemas de información.

Mediante **Ventas/Informes** podemos comprobar que, de los seis componentes que aparecen:

- Los dos primeros los estudiamos en el apartado de compra-venta.
- Mientras que los cinco restantes, van a tomar importancia al realizar el análisis detallado de iniciativas, oportunidades, llmadas, reclamaciones y acciones del soporte (HelpDesk).

## 2. Implantación de sistemas ERP. CRM en una empresa

Cuando comenzamos a trabajar con un proyecto nuevo de implantación, es imprescindibles que realicemos al principio un proceso estructurado y metodológico para conseguir llegar a buen fin el desarrollo, es decir, necesitamos realizar una metodología de implantación.

A lo largo de los años, se han ido desarrollando diferentes estudios para crear planes de instalación para los sistemas informáticos, con sus correspondientes ventajas e inconvenientes. Uno de los principales aspectos que debemos tener en cuenta, desde el primer momento, es saber que una aproximación desorganizada nos va a llevar al fracaso en nuestro proyecto.

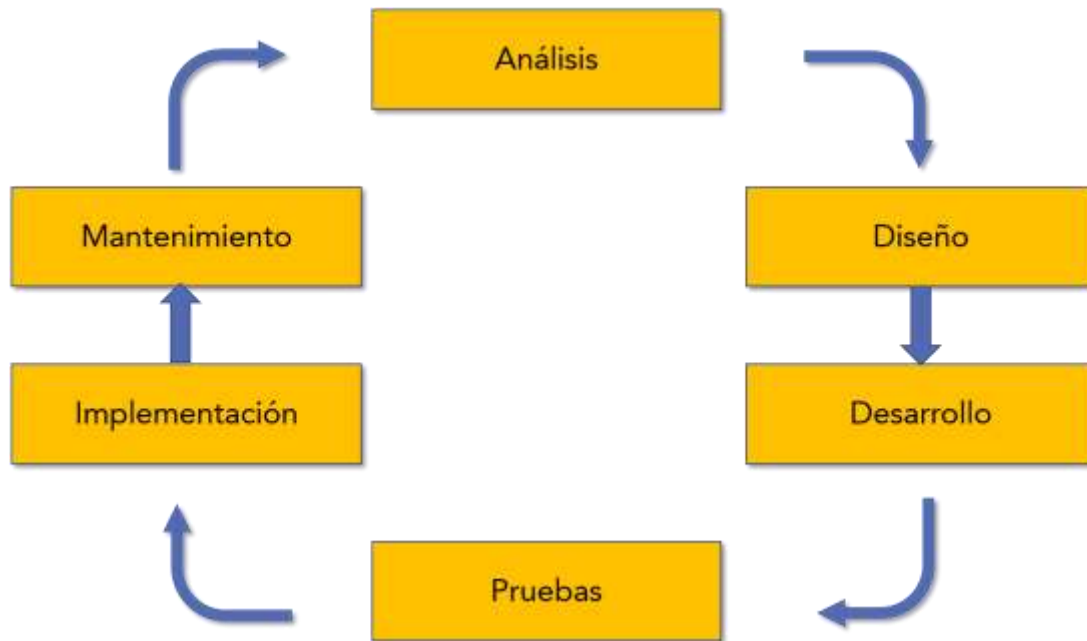
Es importante conocer que, la mayoría de las implantaciones que no llegan a buen fin, son casi siempre por una mala organización. Por tanto, es imprescindible que, establezcamos en un primer momento un procedimiento ordenado y fiable.

**No es posible crear una metodología que sea eficiente para todos los proyectos, ya que cada uno es diferente y tiene sus propias particularidades.** Pero lo que sí podemos llevar a cabo es realizar un **mecanismo que sea común** para todas ellas con posibilidad de que se pueda adaptar a todos los casos que se presenten.

**En este apartado, vamos a definir una metodología que se base en el ciclo de vida tradicional de un determinado proyecto informático, con posibilidad de poder adaptarse a las distintas características de un proyecto determinado (implantación de sistema ERP).**

La idea principal es crear una serie de pasos, haciendo uso de las herramientas más comunes para el personal de informática, generando un tipo de documentación informativa, descriptiva y organizativa que nos ayude a desarrollar la implantación.

Esta propuesta se puede adaptar a las distintas necesidades de un proyecto, sin llegar a ser demasiado exhaustivos. Basta con realizar una simple aproximación para indicar desde dónde vamos a partir.



*Ciclo de vida clásico*

En esta imagen, observamos el **ciclo de vida tradicional** de un proyecto, iniciando en una **fase de análisis**, en la que se deben determinar las principales características del proyecto

Esta fase nos va a llevar a la fase de **diseño** del sistema, tanto a nivel físico como lógico originando la documentación técnica correspondiente para poder llevar a cabo el **desarrollo**.

Una vez que nuestro proyecto está perfectamente diseñado, podemos pasar a desarrollarlo, creando un código determinado para la aplicación y realizando sus **pruebas** correspondientes.

Mediante estas pruebas, podemos asegurar su correcto funcionamiento de todos los módulos correspondientes y, una vez que ya hemos probado el software, es el momento de poderlo **implantar** en la empresa.

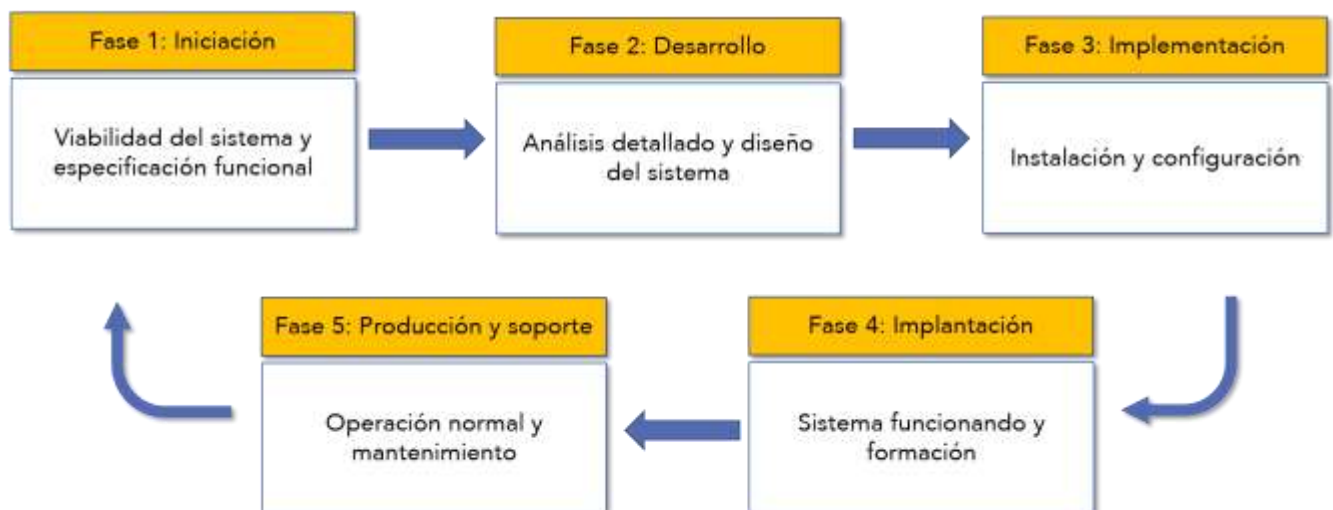
Una vez que se instala, podemos lanzar la fase de mantenimiento que se va a encargar de ir actualizando el sistema, realizando las modificaciones pertinentes.

Intentaremos establecer unas pautas para seleccionar el sistema ERP que más se ajuste a las necesidades de nuestra empresa.

A continuación, veremos las diferentes fases para llevar a cabo la implementación de un sistema ERP en la empresa, ya que tiene que pasar por varios pasos para su implementación.

## 2.1. Metodología de implantación

### ○ Metodología general: Fases



La metodología en la que nos basaremos, va a disponer de **cinco fases**, en vez de seis, como vimos en el apartado anterior.

A continuación, las detallaremos más detenidamente:

- **Fase 1: Iniciación** Su función principal va a ser, determinar al más alto nivel, una serie de propiedades que necesita el cliente junto con el tiempo estimado para llevarlo a cabo. Por tanto, se deben sentar las bases para mantener el proyecto bajo control y garantizar que el sistema puede satisfacer las necesidades que se le exigen. Dentro de esta fase, podemos diferenciar cuatro puntos, que especificaremos a continuación:

- **Estudiar el ámbito del proyecto:** Para poder llevar a cabo esta fase y conocer el alcance y ámbito de la empresa, tendremos que realizar una serie de reuniones. En ellas se determinará las funciones a realizar por el sistema y la forma de proceder. Se planificará el planteamiento para llevar a cabo la implantación de la aplicación con los distintos departamentos y se regulará las acciones que deben realizar cada uno de ellos. Como consecuencia de todas las reuniones, realizaremos un documento con todo lo hablado en ella y la planificación temporal del proyecto.
- **Realizar estudio de viabilidad económica, técnica y organizativa:** A partir del documento anterior, habrá que estudiar la viabilidad de todas las necesidades expuestas y la prioridad correspondiente. En esta fase debemos de estudiar la solvencia de la propuesta y las modificaciones que se harán en el futuro. Como solución a esta fase, debemos de responder con la continuación del proyecto o la desestimación por completo. En ambos casos debemos redactar un informe con las causas de la desestimación o el contrato con las cláusulas para que el departamento legal lo revise.
- **Determinar nivel de cambio a un nuevo sistema:** El paso siguiente y tras los informes realizar en la anterior fase, si la respuesta ha sido afirmativa, debemos de ponernos en marcha para implantar el nuevo sistema. Para ello debemos de contar con los técnicos pertinentes y mediante reuniones llegar a detallar cada paso, conociendo los elementos involucrados, las medias de seguridad a tener en cuenta, demás riesgos que pueden ir surgiendo. Ante cualquier incidencia debemos de detallar los pasos para que se solvente los más pronto posible. Como es de costumbre, realizaremos un documento con los detalles, medidas preventivas y riesgos a asumir en dicha fase.
- **Organizar el proyecto y planificarlo:** Para terminar esta fase de organización, lo único que debemos de razonar son las herramientas a utilizar para llevar a cabo. Además, debemos de contar con la planificación del proyecto, para ello podemos hacer uso de una herramienta de planificación de proyecto. De



esta forma diseñaremos un documento funcional del sistema y un plan de ayuda para llevar a cabo el proyecto.

- **Fase 2: Desarrollo:** En la fase de desarrollo podemos crear el sistema informático completo sobre papel. Comenzaremos analizando la especificación funcional y, de esta forma, ampliar el documento (mediante diseño top- down) hasta el máximo nivel en el que se especifique la entrada- salida de datos, modificaciones de los mismos, etc.

Una vez que el sistema esté descrito mediante DFD, podemos pasar a crear el diseño hardware del sistema: ordenadores, conexiones, distribución, etc. Y, seguidamente, podemos definir todos los elementos software que sean necesarios: módulos a utilizar, permisos, configuraciones, etc.

Estas subfases las podemos resumir en:

- Análisis detallado
  - Diseño físico del sistema
  - Diseño lógico del sistema
  - Revisión de previsiones
- **Fase 3: Implementación:** Cuando lleguemos a esta fase de implementación, querrá decir que ya tenemos completamente definido nuestro sistema, por tanto, podemos comenzar a instalarlo.

Lo primero que debemos hacer es adquirir el hardware necesario para tal fin y, a continuación, podemos comenzar a crear la red subyacente para poder configurar de forma completa todo lo adquirido. Además, podremos instalar todo el software relativo al sistema de gestión para configurarlo de forma adecuada y poder añadir los módulos que sean necesarios para comenzar con las diferentes pruebas.

Esta fase debe documentarse de forma completa desde su inicio, asignando al equipo de mantenimiento que después se va a encargar

de la información suficiente y detallada en el sistema. Por lo que las subfases en las que lo dividimos quedarían de la siguiente forma:

- Adquisición del hardware
- Desarrollo de software
- Plan de pruebas
- Documentación
  
- **Fase 4: Implantación:** Llegados a la fase de implantación, es el momento de incorporar el sistema gestor a la empresa, asumiendo el control de todas las funciones especificadas en el contrato. Debemos comprobar que todo es correcto.

Para realizar esta fase de implantación de forma correcta, debemos formar a todos los usuarios para ofrecer soporte inicial, supervisar todo el funcionamiento del sistema, para adaptar los datos antiguos a las nuevas situaciones. Finalmente, debemos comprobar que todo tiene un funcionamiento correcto mediante una serie de pruebas que va a realizar el cliente para, poder demostrar así que el trabajo está terminado.

Estas fases, se dividen en:

- Plan de implantación.
- Implantación.
- Formación.
- Conversión y migración de datos.
- Test de aceptación.
  
- **Fase 5: Producción y soporte:** Si estamos en la fase cinco, debe ser porque el proyecto ha concluido su tarea, es decir, ha finalizado cumpliendo sus expectativas iniciales en el tiempo estimado. Ya en las entrevistas finales es cuando vamos a entregar todo el material disponible al cliente, recopilando toda la información hasta el momento.

A pesar de que esta es la última fase en nuestro proceso, debemos señalar que, a partir de este momento, comienza un largo proceso de operación, mantenimiento y soporte al sistema.

Mediante la operación y soporte, nos aseguraremos de que los procesos siguen funcionando y que los usuarios están haciendo un buen uso de él. Y, a través del mantenimiento nos aseguraremos de que las modificaciones que se realizan al sistema son para solventar algún error que haya podido surgir o porque existan nuevas necesidades.

Debemos llevar a cabo estos cambios sin que afecte al resto del sistema.

Nuestra función en esta fase, va a depender bastante del **tipo de contrato** que tengamos:

- Si no tenemos contratado el mantenimiento y soporte daremos por finalizada nuestra función, llevando a cabo una auditoría del sistema transcurridos unos meses después de comprobar el funcionamiento del mismo. En esta auditoría debemos especificar aquellos fallos que se hayan detectado y corregidos en este tiempo además de las diferentes modificaciones que se hayan producido. El objeto principal es detectar la corrección de nuestro trabajo.
- Si tenemos especificado en el contrato la implementación de los mecanismos de modificación y resolución de problemas, debemos comprobar los errores que se hayan podido producir en este tiempo y resolverlos en el menor tiempo posible. Es importante que sepamos la cantidad de errores producidos para saber cómo se está comportando nuestro sistema.

Estas fases, se dividen en:

- Operación normal.
- Soporte.
- Mantenimiento.
- Documentación al cliente.

## 2.2. Tipo de empresa. Necesidades de la empresa.

Una aplicación ERP es un software que ayuda a la gestión diaria de la empresa. Desde la planificación y control de las tareas a realizar hasta los recursos disponibles en la empresa. Tiene el objetivo de conseguir que todos los datos importantes de la compañía estén integrados en una misma herramienta, sin tener en cuenta al área a la que pertenece la información.

Contar con un ERP como sistema de gestión empresarial facilita enormemente la trazabilidad de las operaciones y con ello la resolución rápida de problemas. Todo esto se ve reflejado en la respuesta rápida al cliente ante cualquier pregunta.

Debemos de concienciarnos que el concepto de ERP va más allá del software, ya que debe reflejar los procesos de una empresa y cuanto más adaptable, esté la herramienta, mejor servicio podremos dar a nuestra clientela ya que nos hará funcionar de una forma más óptima.

### ○ Adaptación del ERP a una empresa

Una de las principales tareas del análisis y diseño es saber detectar e implementar los diferentes cambios en el sistema OpenERP para poderlo adaptar a la empresa actual. Podemos realizar esta adaptación en tres ámbitos diferentes:

- Estableciendo nuevas vistas con información existente en la base de datos (Informes)
- Modificando y creando nuevos informes adaptándolos a la empresa.
- Programando módulos completos que añadan todas las características necesarias.

## 2.3. Selección de los módulos del sistema ERP- CRM.

### ○ Pasos para crear un módulo

Cuando trabajamos con módulos, es muy importante que comprendamos qué es un módulo, cuáles son los elementos que lo forman y cómo se crean.

Por tanto, vamos a crearnos un módulo para dar solución a estas cuestiones.

1. Nos posicionamos en el directorio **addons** que se encuentra dentro de la instalación del Server y, creamos un nuevo directorio para nuestro módulo: **mimodulo**

2. A continuación, añadimos los ficheros obligatorios:

`_init_.py`

### **\_terp\_py**

3. Ya podemos crear el fichero de definición del objeto **mimodulo.py** vacío.
4. Añadimos el fichero **mimodulo\_view.xml** que va a contener los diferentes datos para el sistema.
5. Una vez programado el contenido de los ficheros, debemos abrir una sesión con el servidor, por lo que nos desplazamos hasta **Administración/Módulos** para actualizar la lista de módulos.

Una vez que se detecte el módulo, ya lo podemos instalar:

- **El fichero \_init\_.py:**  
Lo utilizaremos para cargar en el sistema ñas definiciones de nuestros objetos.
- **El fichero \_terp\_.py:**  
Define un diccionario anónimo con las propiedades ya definidas que va a utilizar el sistema para determinar aquellos ficheros XML que se deben seguir tratando junto con las propiedades correspondientes a cada módulo y sus dependencias con los demás módulos del sistema.

Los valores que puede utilizar suelen ser:

- **name.** Para indicar el nombre del módulo. Es un campo obligatorio.
- **autor.** Nombre del autor.
- **version.** Número de la versión del módulo que se irá incrementando de uno en uno.
- **description.** Describe la función del módulo.
- **website.** Url correspondiente a la dirección del programador.
- **depends.** Lista que indica los módulos que deben estar instalados para que éste pueda funcionar.
- **init\_xml.** Lista de ficheros XML que se van a utilizar cuando lancemos el servidor.
- **update\_xml.** Lista los ficheros que se van a utilizar cuando se lance una actualización del módulo.
- **category.** Descripción de la categoría y subcategoría del módulo.

- **active.** Corresponde a un valor booleano (True o False) que nos va a permitir determinar si un módulo se debe instalar mediante la creación de la base de datos o no.
- **installable.** Va a determinar si el módulo se puede instalar o no, mediante los valores de True o False.
- **demo\_xml.** Lista correspondiente a los ficheros XML con los datos de prueba.
- **license.** Hace referencia a la licencia del módulo.

○ **Fichero mimodulo.py:**

Mediante este fichero podemos determinar aquellos objetos nuevos, junto con sus propiedades, en formato Python.

También podemos utilizar herramientas tipo DIA para la creación de diagramas de clases y la generación de código automática.

En OpenERP, todo lo que se encuentra almacenado en la base de datos son objetos y, hace uso de un mecanismo de ORM (Mapeo de Objetos Relacionales), para mover las tablas de objetos relacionales a la programación. Así, cuando creamos un objeto, se va a crear en este fichero, de forma automática, una tabla que se encuentre asociada a los atributos que definamos.

**OpenERP utiliza una estructura jerárquica, permitiendo crear un objeto en cada nivel.**

En el interior de un objeto, una clase, debemos utilizar, de forma obligatoria, una serie de propiedades que nos van a permitir especificar los principales componentes del mismo, como pueden ser:

- **\_name.** Nombre correspondiente al objeto, la tabla a crear.
- **\_columns.** Campo obligatorio que va a contener la definición de los distintos campos de la tabla.
- **\_constrain.** Restricciones de los campos.
- **\_defaluts.** Valores de los campos por defecto.
- **\_inherit.** Objeto del que se hereda.
- **\_order.** Campos que se pueden utilizar en las diferentes sentencias de selección y lectura.

- **\_rec-name.** Campo que podemos utilizar cuando deseemos realizar alguna búsqueda.
- **Fichero mimodulo\_view.xml**

Este fichero nos va a permitir determinar los distintos componentes OpenERP que debemos crear para conseguir que nuestro módulo se integre con el sistema. Un determinado módulo, va a estar formado por vistas, gráficos, informes, asistentes, flujos de trabajo, menús y acciones.

A continuación, detallaremos algunos de ellos de forma más específica.

- **Vista.**

Es una representación gráfica del objeto en la parte del cliente. OpenERP va a utilizar el paradigma MVC (**Modelo Vista Controlador**) para diferenciar los datos de su representación. Mediante Python, hemos definido la estructura de los distintos datos y, en este, debemos definir la estructura de visualización.

Podemos diferenciar **dos tipos de vistas**:

- Tipo árbol: cuando listamos el contenido por filas.
- Tipo formulario: cuando tenemos que modificar e insertar datos.

- **Menú.**

Elemento que vamos a utilizar para indicar, en la parte cliente, una determinada acción, como pueden ser, abrir una determinada ventana, abrir informes, etc.

- **Informe.**

Recolección de datos en formato PDF o HTML.

- **Gráfico.**

Representación gráfica de un conjunto de datos.

- **Asistente.**

Conjunto de los distintos pasos secuenciales que debemos seguir a la hora de configurar un módulo.

- **Flujo de trabajo.**

Definición dinámica de los datos: cómo se crean, cómo se modifican, etc.

Una vez que creamos un módulo, vamos a necesitar, como mínimo, la definición de los menús además de las acciones de apertura de una vista, vistas de árbol y formulario.

Sin ellas es muy difícil instalar un módulo correctamente.

## 2.4. Tablas y vistas a adaptar.

Una herramienta ERP hace uso del Modelo- Vista- Controlador para llevar a cabo la gestión de datos. Cuando separamos los datos de la presentación, podemos conseguir la independencia, pudiendo llevar a cabo la implementación de los distintos entornos de acceso al servidor.

La estructura que se utilizar para almacenar la información es la misma que en las bases de datos, es decir, las tablas. Es más, una herramienta ERP, tiene como motor una base de datos ya que es la mejor manera para organizar, relacionar y almacenar los datos con los que después podemos manejarlos

Las interfaces que se le presentan a los clientes, mediante ficheros XML, son dinámicas, por lo que se pueden editar en cualquier momento. Si algún fichero cambia, sólo debemos volver a cargarlo para actualizar las modificaciones oportunas.

Una vez llegado el momento de diseñar los diferentes modelos de representación, debemos determinar cómo se va a realizar la visualización por pantalla. Por tanto, podemos diferenciar entre dos tipos de vistas principales más otra de apoyo.

### ○ Vista formulario

Utilizada cuando editamos los datos, los campos se pueden distribuir a lo largo de toda la visualización, utilizando una serie de reglas, como pueden ser:

- Cada campo debe tener su etiqueta con el nombre.
- Los campos van situados de izquierda a derecha y de arriba abajo dependiendo del orden en el que estén definidos.
- A cada pantalla la definen cuatro columnas y un número de filas indeterminado. Cada campo va a utilizar, por defecto, dos columnas, una para la etiqueta y la otra para el campo.



- Podemos utilizar una etiqueta especial para dividir una columna en tantas como necesitemos.
- **Vista en árbol**

Utilizadas en los listados de datos ya que son bastante sencillas y disponen de menos opciones.

- **Vista de búsqueda**

Sirve para complementar a la vista en árbol ya que permite añadir un panel de búsqueda y filtrado en su parte superior.

- **Vista de gráfico**

Es una nueva forma de vista para que los distintos formularios puedan mostrar un gráfico que ha sido formado a partir de unos datos determinados.

- **Definición de las vistas**

Las vistas se van a definir en el fichero **XML** que vayamos a establecer en el fichero **\_terp\_.py**, ya que posee una estructura especial que necesita, al menos de tres elementos para poder definirla de forma correcta. Estos elementos que debemos crear, pueden ser:

- Una **acción** para realizar una vista.
- Un **menú** que sirva para poder ejecutar la acción.
- Un conjunto de **vistas**, que van a estar asociadas a la acción.

Es conveniente que comencemos por la acción y vamos a utilizar la etiqueta **<record model="ir.actions.act\_windows">** para definirla.

Seguidamente, podemos definir el menú que tiene asociado, mediante la etiqueta **<menuitem>**. Finalmente, podemos crear tantas vistas como etiquetas tengamos **<record model="ir.ui.view">**.

Dentro de la etiqueta de definición de vista (tree, form, serach o graph) debemos crear la estructura para visualizar adecuada para el usuario. En el momento de implementarla, podemos diferenciar entre dos tipos de elementos: **grupales** (los que no muestran datos) y, **de datos** (los que organizan los datos).

- **Grupales**

- **<separator string="Nombre a mostrar" colspan="Número">**

Permite crear una división entre los diferentes campos mediante una línea y un texto. El texto que debemos mostrar va a aparecer en la propiedad **string** y el número de columnas necesarias se van a definir mediante **colspan**.

- **<notebook colspan="Número">**

Permite crear un control de las distintas pestañas. Como mínimo, vamos a disponer de una pestaña dentro que la definiremos con **<page**.

- **<page string="">**

Se trata de una etiqueta que nos va a permitir organizar en pestañas el contenido de un notebook. Podremos crear tantas etiquetas como pestañas sean necesarias y dentro, podremos definir la visualización haciendo uso de otras etiquetas específicas de agrupación, salvo notebook y page.

- **<group colspan="Número" rowspan="Número" expands="yes" col="Número" string="Cadena a mostrar">**

Nos va a permitir agrupar distintos controles de datos en columnas de la siguiente forma:

- a. Primero, el control de puede extender a través del número de columnas que están especificadas en **colspan** y de filas con **rowspan**.
- b. Después, podemos dividir el espacio asignado en columnas (las que estén especificadas en la propiedad **col**).
- c. Y, por último, el espacio que no utilizemos, se puede distribuir entre el resto siempre y cuando la propiedad **expand** tenga como valor, yes.

- **Elementos de datos**

- **<newline />**

Nos permite crear un salto de línea en la visualización, de tal forma que obliga al siguiente control a pasar a la siguiente fila.

- **<label string="Texto" />**

Nos permite añadir una etiqueta con su texto correspondiente.

○ **<field>**

Define un campo que tiene que estar definido en el modelo de datos obligatoriamente. Las propiedades que tenemos disponibles son:

`name=""`.

Corresponde al campo que está definido en el fichero Python en una clase correspondiente al objeto que vayamos a utilizar en la definición de la vista.

- **`select="1"`**. Cuando deseemos que el campo actual sea un campo que se utilice en búsquedas, por lo que debe tener un índice.
- **`colspan="Numero"`**. Hace referencia a la cantidad de columnas que podemos utilizar por parte del campo, por defecto son dos con la etiqueta incluida.
- **`readonly="1"`**. Si queremos el que campo sea de sólo lectura.
- **`required="1"`**. Es un campo obligatorio incluso si no está así definido en el modelo de datos inicial.
- **`no label="1"`**. No imprime la etiqueta asociada al nombre del campo.
- **`invisible="True"`**. Oculta el campo y su correspondiente etiqueta.
- **`string="Texto"`**. Es el valor que vamos a asignar a la etiqueta en caso de que no se utilice el nombre del campo.
- **`onchange="funcion"`**. Nombre y parámetros de la función que tenemos que llamar cuando cambia el valor del campo.
- **`widget="tipo"`**. Permite cambiar por otro el control por defecto que se va a visualizar.

○ **Las vistas gráficas y de búsqueda**

La vista gráfica nos va a permitir crear un gráfico de barras o circular partiendo de los campos que determinemos. En este tipo de vista vamos a poder realizar agrupaciones sobre los campos para, conseguir así poder realizar diferentes operaciones simples sobre los grupos, como pueden ser las sumas, multiplicaciones, etc.

La vista gráfica es de tipo árbol y va a representar el panel de búsqueda en la parte superior.

Las vistas de búsqueda, cuentan con la posibilidad de poderse añadir a las de árbol y gráficas para, de esta forma, realizar una serie de operaciones de filtrado complejas. Se incorporarán aquellos campos necesarios de forma normal mediante el uso de una etiqueta **<field**, teniendo la posibilidad de poderlos agrupar con la etiqueta **<group**.

Si queremos añadir botones de agrupación, podemos utilizar una entrada **<filter** en la que definimos, mediante la propiedad **context** el campo que deseemos agrupar.

### ○ Herencia en las vistas

Al definir las vistas de aquellos objetos heredados, se nos pueden plantear una serie de posibilidades:

La posibilidad de crear una nueva vista completa desde cero, referenciando al nuevo modelo o volver a utilizar la vista que ha creado el padre y volver a definirla.

Cuando redefinimos una vista, debemos tener en cuenta que el objeto que acabamos de crear ha extendido la tabla del padre para conseguir añadir campos, no hemos creado una nueva.

Una vez que ya hemos determinado los campos que vamos a utilizar, debemos decirle a la vista que utilice la que está definida por el padre. Haremos esto cuando podamos añadir en la definición de la vista el campo **<field name="inherit\_id" ref="id\_vista\_padre"/>**. Así el sistema va a interpretar la vista a la que referenciamos, incorporándola a nuestra definición automáticamente.

Con esta incorporación de la vista padre, vamos a conseguir que todos los campos existentes se añadan a la nueva para que, a continuación, realicemos las modificaciones oportunas para agregar nuevos campos.

Una vez llegado el momento de añadir nuevos campos a la vista heredada, debemos seguir una sintaxis especial que debemos tener en cuenta:

- La etiqueta **<form** cambia por **<data**
- Si deseamos eliminar algún campo, debemos añadir una etiqueta
  - **<field name="nombre\_campo\_eliminar\_de\_la\_vista\_padre" position="replace"/>**
- Si lo que pretendemos es reemplazar un campo por otro, añadiremos la etiqueta **<field name="nuevo campo">**

- Para definir los campos debemos seleccionar un campo base que exista previamente a través de `<field` para añadir dentro tantas etiquetas como campos nuevos deseemos crear.
- En los casos en los que sea más complicado determinar un campo porque exista varias veces, utilizaremos un elemento `<xpath` de XML para realizar una búsqueda sobre el árbol creado y realizar en su interior todas las modificaciones correspondientes.

## 2.5. Consultas necesarias para obtener información.

Este sistema ERP al estar basado en motores de Bases de datos, presenta la misma dinámica de trabajo que un sistema gestor de Bases de Datos.

Una vez almacenada la información en las tablas, una de las ventajas que presenta este sistema es que, al estar la información relacionadas entre sí, podemos consultar datos referidos a tablas relacionadas. De esta forma sacaremos más jugo a la información almacenada, hasta tal punto de conocer algunos puntos importantes para ponerlo en práctica para el futuro.

Este aspecto es interesante para el tema de campañas de marketing y conocer el área de cliente al que va dirigido.

## 2.6. Creación de formularios personalizados.

La interfaz gráfica destinada a visualizar los datos de una manera clara y atractiva para que el usuario final, pueda ver la información de una forma más atractiva y dinámica. Además, tiene otra utilidad la utilización de los formularios, la entrada de datos se hace de una forma más atractiva y fácil para el usuario.

Cada herramienta ERP, tiene sus formas de crear formularios y de personalizarlos, pero todas tienen en común la facilidad para realizar dicha acción. Normalmente se cuenta con un asistente que nos ayudará a diseñar el formulario y todos los campos que necesite dicho formulario, e incluso insertar un subformulario dentro de otro.

Dicha herramienta de ayuda para crear formularios, estará diseñada para que de una forma gráfica e intuitiva el usuario fácilmente puede personalizarlos y además ejecutarlos.

## 2.7. Creación de informes personalizados.

Hacen referencia a una serie de nuevas vistas que hemos creado para almacenar la información correspondiente a la base de datos ofreciendo la posibilidad de poder gestionarla de manera más cómoda y práctica.

Estos informes no van a incorporar estructuras novedosas, sólo va a utilizar las existentes aumentando de esta forma su eficiencia.

A continuación, vamos a indicar los diferentes **pasos que debemos seguir para crear una nueva vista**:

1. **Configuramos el sistema.** Debemos lanzar el asistente de instalación y configuración que se ejecuta cuando lanzamos el proyecto, a través de **Administración/ Configuración/Iniciar la configuración**.

Y una vez que se muestre el asistente, seleccionaremos la opción de **Informes avanzados**. A continuación, seleccionamos **Instalar** para comenzar con el proceso. Iremos saltando todas las ventanas que aparezcan excepto la de **Configura las herramientas de informes**.

En esta ventana, debemos seleccionar las opciones de **Constructor de consultas** y la de **Diseñador de informes OpenOffice** y, a continuación, seleccionaremos la opción de **Configurar**.

2. **Una vez llegados a este punto, el sistema ya está configurado**, de tal forma que, podemos pasar a crear una nueva vista (informe). Para ello, seleccionaremos la opción de **Administración / Personalización / Informes / Informes personalizados**.

Debemos seleccionar el objeto base del sistema que vamos a utilizar en la pestaña de **Configuración general**. Seguidamente, podemos definir los campos que sean obligatorios para determinar los datos que sean básicos mediante la pestaña de **Parámetros de la vista**.

3. Podemos fijar aquellos campos que vamos a utilizar en la pestaña **Campos a mostrar** estableciendo para el campo **secuencia** un determinado valor numérico.

4. Para finalizar, tenemos la posibilidad de establecer filtros sobre alguno de los campos seleccionados en la pestaña **Filtros en campos**.

5. En este punto ya podemos finalizar el diseño del informe por lo que podemos seleccionar la opción de **Guardar**.

Nos faltaría comentar todos aquellos aspectos importantes que debamos señalar. Toda la información se va a almacenar en una base de datos relacional, por lo que un informe de esta forma no es más que una consulta SELECT que se realiza a dicha base de datos para mostrar un resultado determinado.

6. Llegado el momento de lanzar la vista, lo llevaremos a cabo mediante la pestaña **Configuración general** a través del botón **Abrir informe**.

### ○ Tablero personalizado de informes

Los informes personalizados cuentan con una forma bastante organizada de manejar un conjunto de datos. De la misma forma, los tableros pueden organizar diferentes informes para conseguir mejorar la productividad.

Para poder explicar este apartado y tras haber creado varios tableros, de forma estándar, ya podemos definir los nuestros para utilizarlos.

- Si deseamos crear un tablero nuevo, seleccionaremos **Administración/Personalización/Informes/Definición de tablero**. De esta forma, cada entrada se va a asociar con una acción diferente (informe) y así poder establecer las distintas características de presentación. Podemos tener tantos componentes como sean necesarios.
- Una vez definido el tablero, podemos crear una entrada en el menú para poder acceder a él.
- Cuando lancemos el asistente desde **Crear menú**, nos va a preguntar sobre el nombre del objeto padre para posicionar la nueva entrada junto con el nombre a mostrar.

### ○ Creación de informes impresos

OpenERP cuenta con una serie de mecanismos que podemos utilizar para crear informes imprimibles haciendo uso de la información almacenada en la base de datos. Entre los más conocidos, podemos señalar JasperReports, motor Aeroo, programación directa de RML y OpenOffice.

En este caso, utilizaremos OpenOffice para crear nuestros informes, subirlos al servidor, etc.

1. Lo primero que vamos a hacer es instalar OpenOffice y configurarlo para los informes. Comenzaremos desde la base de datos que tenemos instalada, abrimos OpenOffice Writer y seleccionamos la opción de **Herramientas/Gestión de extensiones**.

Haciendo uso del botón **Añadir**, podremos buscar el fichero que guardamos en la configuración inicial del sistema. Esta acción nos va a permitir cargar y configurar OpenOffice a la hora de crear informes. Es conveniente que cerremos la aplicación y reiniciemos para que vayan apareciendo los nuevos elementos.

2. Para crear los informes, debemos configurar antes las opciones de conexión con el servidor, teniendo en cuenta que, sólo el administrador va a tener permisos para subir informes utilizando este mecanismo.

3. Cuando comencemos a crear los informes, podemos hacerlo de dos formas distintas:

- Partir desde cero para crearlo nosotros.
- Crear un informe partiendo de las modificaciones que realicemos a alguno ya creado. En este caso utilizaremos la opción de **OpenERP Report Designer/Modify Existing Report** para obtener un listado de todos los informes, seleccionaremos el que deseemos y, a continuación, realizaremos las modificaciones deseadas.
- Una vez que terminemos de modificarlo, tenemos que grabarlo en el disco con un nombre y lo enviaremos al servidor, mediante **OpenERP Report Designer/ Send to the server**

4. Para este ejemplo, nosotros vamos a seleccionar la opción de crear un informe nuevo. Seleccionamos **OpenERP Report Designer/Open a new report** para que nos muestro todos los módulos que podemos utilizar como base para realizar nuestro documento.

5. Lo siguiente que debemos crear es el conjunto de elementos que deseemos mostrar, por lo que seleccionamos una lista con todos los campos disponibles. Accederemos a las listas mediante la opción del menú **OpenERP Report Designer/Add a loop**

De todas las listas que aparecen, seleccionamos la que más nos interese en cada momento.

6. Una vez que seleccionamos el conjunto de elementos que deseemos mostrar, podemos comenzar a crear la estructura del informe, escribiendo el texto correspondiente, insertando imágenes, etc.

En un campo de la base de datos, debemos añadir el botón correspondiente al menú **OpenERP Report Designer/Add a field**



7. Una vez ya diseñado el informe, tenemos que grabarlo en disco y asignarle un nombre.

8. Por último, tenemos que subirlo al servidor mediante **OpenERP Report Designer / Send to the server** asignando el nombre correspondiente al informe junto con el tipo de informe que se va a generar.

9. Ya podemos acceder al informe que va a aparecer, por defecto en la pestaña de la parte lateral derecha del objeto que hemos seleccionado.

Cada vez que seleccionemos el informe, descargaremos el fichero que se ha generado accediendo a la primera página que se ha generado.

Comprobaremos que todo es correcto accediendo al servidor en el elemento **Administración / Personalización / Objetos de bajo nivel / Acciones / Informes**

## 2.8. Creación de cuadros de mando personalizados.

Al ser una herramienta con una base en código fuente libre, está generalizado a cualquier tipo de empresa, ya que cubre las necesidades empresariales más generales. Además de esta apariencia estándar, podemos desarrollar a partir de ella, un interfaz y una base de datos más acorde a las necesidades específicas de cada corporación, y que el software se adapte lo máximo posible a la forma de actuar de cada empresa. De esta forma, tendremos un software realizado a nuestra medida con muy poco desarrollo. Esta es una de las ventajas que proporciona esta tecnología y por esta razón su gran auge a la hora de implantarla en el ámbito empresarial.

Para llevar a cabo este proceso de adaptación del código a la empresa, debemos contactar con personal experto del área de la informática. Otra de las ventajas de este software a medida es la puesta en marcha, ya que el periodo de implantación en la empresa no tendría coste alguno, ya que está adaptado 100% al protocolo de actuación laboral.

### 3. Desarrollo de componentes.

En este tema de desarrollo de componentes, nos vamos a centrar en el lenguaje Python, en la versión 2.7. Seleccionamos este lenguaje porque OpenERP, sistema que hemos visto en la Unidad Formativa anterior, está programado bajo Python, por lo que es imprescindible que conozcamos este lenguaje de programación y, de esta forma, poder programar diferentes módulos para el sistema en cuestión.

Dividiremos este proceso en dos partes:

1. En la primera, nos centraremos en el lenguaje que vamos a utilizar, aunque no entraremos en detalles más específicos como pueden ser el acceso a la red por parte de Python, el tratamiento sobre los datos HTML y XML, etc. Para ello, se recomienda algún libro específico sobre el lenguaje en cuestión.
2. Y en la segunda, entraremos un poco más en la creación de módulos propios que podemos integrar en un sistema Open ERP funcional.

Optamos por la versión 2.7 porque OpenERP utiliza esta versión en su interior.

En una de las partes nos centraremos en la implementación de los diferentes módulos para OpenERP, comprobando la estructura de ficheros que necesitamos, su forma de implementarlo y su correspondiente instalación en el servidor.

Veremos una serie de ejemplos sobre distintos módulos para comprobar aspectos determinados sobre la programación. Antes de comenzar a programar los módulos, realizaremos el análisis y diseño correspondiente.

**Debemos analizar el módulo para definir el diseño de los datos, su almacenamiento, las distintas relaciones, los posibles procedimientos, etc.**  
**Para este análisis, vamos a utilizar las técnicas y estándares que necesitemos para asegurarnos de qué es lo que debemos hacer y cómo lo haremos.**

### 3.1. Lenguaje proporcionado por los sistemas ERP- CRM. Características y sintaxis del lenguaje. Declaración de datos. Estructuras de programación. Sentencias del lenguaje.

- Lenguaje de programación Python

Este lenguaje de programación, surge alrededor de los años 90, con la intención de poder programar en los distintos servidores web, de forma limpia y multiplataforma.

Este lenguaje, en principio, pretendía ser interpretado y orientado a objetos, añadiendo los paradigmas de la programación tradicional.



- Comenzando con Python

Como ya hemos indicado, Python es un lenguaje que puede ser interpretado multiplataforma, convirtiendo su código fuente en objeto cuando se ejecuta por primera vez. De esta forma, consigue juntar lo mejor de estos dos mundos, utilizando la flexibilidad de los lenguajes script con la velocidad de un lenguaje ya compilado.

**Tenemos la opción de no declarar las variables que vamos a utilizar, aunque sí que es obligatorio, asignarle algún valor para inicializarlas antes de que la utilicemos.** Como no se determina el tipo que se puede almacenar en una variable, ésta va a poder adoptar un valor cualquiera para cada momento, de tal forma que se va a determinar cuándo se esté ejecutando según el contenido que tenga. Las variables no se pueden convertir, de un tipo a otro de forma automática, ya que, para tal fin, necesitan realizar un casting.

Al ser un lenguaje orientado a objetos, cuenta con la opción de definir todo como un objeto, ofreciendo la posibilidad de crear diferentes clases y objetos, implementar la herencia sin límite de acceso a las propiedades o funciones definidas.

En el lenguaje Python no existen mecanismos para definir bloques explícitos, sino que se van creando con la implementación del propio código. Los

bloques comienzan con (:) y todo lo que lleva dentro, debe contener una serie de espacios en blanco como sangrado. Es muy importante la cantidad de espacios en blanco que utilicemos ya que deben coincidir en todas las líneas que compongan el bloque. En caso de no coincidir, dará error. Es un fallo bastante común al confundir el tabulador con el espacio ya que el lenguaje lo interpreta de diferente forma.

PYTHON	JAVA
<pre>edad=1 while edad&lt;18:     edad+=1     if (edad%2)==0:         continue     print "Es impar," print "Adios"</pre>	<pre>edad=1 while (edad&lt;18){     edad++     if (edad%2==0){         continue     }     System.out.println("Es impar") } System.out.println("Adios")</pre>

Es importante señalar que disponemos de distintas implementaciones para Python dependiendo del lenguaje base que se utilice:

- **CPython**: está escrito en C.
- **JPython**: está escrito en Java.
- **IronPython**: está escrito en C#.

De todas las anteriores, decir que la más utilizada es la versión de C ya que es la que ofrece más estabilidad y madurez.

- **Tipos básicos**
  - **Representación numérica**

Podemos utilizarla mediante enteros simples (3) y enteros largos (3L), siendo la diferencia entre ellos, la cantidad de valores que pueden almacenar. También podemos utilizar la notación tradicional en octal (027) y hexadecimal (0x3F), decimales o en coma flotante (23.45) y en números complejos (2+7i). Lo más importante es que debemos saber cuál es el valor máximo según el tipo que utilicemos y la plataforma en la que ejecutemos.

- **Coma flotante**

La utilizaremos para la representación de valores decimales, teniendo en cuenta la menor precisión que garantiza Python que viene expresada en el estándar IEEE 754 que determina que los valores deben estar comprendidos entre  $\pm 2.22 \cdot 10^{-308}$  y  $\pm 1.79 \cdot 10^{308}$ .

Para aquellos casos en los que necesitemos una mayor precisión, es recomendable que utilicemos valores en Decimal. En este caso, es un objeto específico de Python por lo que lo debemos importar. Nos ofrece una precisión de hasta veintiocho decimales. Los números decimales, también los podemos escribir mediante notación científica si lo preferimos. Y los números complejos, que se dividen en dos partes, la real y la imaginaria, también pueden operar entre ellos.

- **Operadores numéricos básicos**

Entre los operadores básicos, podemos encontrar:

**suma (+), resta (-), multiplicación (\*), división (/), división entera (//),  
exponenciación (\*\*) y módulo (%).**

La división y la división entera no son diferentes siempre que los dos operandos sean enteros.

Para el tratamiento de los bits, podemos utilizar los siguientes operadores:

**Operación and (&), operación or (|), operación xor (^), operación not (~),  
desplazamiento a la izquierda (<<) y desplazamiento a la derecha (>>)**

**NOTA**

Como ya hemos indicado, todos los elementos, para Python, son objetos, así que los tipos numéricos, también lo son.

Los números, además, cuentan con un conjunto de funciones que ofrecen el acceso a diferentes funciones matemáticas avanzadas, como pueden ser, entre otras:

- La raíz cuadrada (**sqrt**)
- Logaritmos (**log10**)

- **Tipo Booleano**

Es el más conveniente cuando trabajamos con expresiones condicionales. Sólo permite almacenar los valores de True o False. El tipo booleano, define una serie de operando, como pueden ser:

- Comparación y → (and)
- Comparación o → (or)
- Negación de una expresión → (not)
- Igualdad → (==)
- Desigualdad → (!=)
- Mayor → (>)
- Mayor igual → (>=)
- Menor → (<)
- Menor igual → (<=)

- **Tipo cadena**

Podemos representar las cadenas mediante comillas simples o bien, comillas dobles, nunca combinando las dos formas a la vez.

Mediante la barra invertida (\) puede introducir una serie de valores especiales, como pueden ser, entre otras:

- \n → Para representar el carácter de una nueva línea.
- \t → Para tabular
- \r → Representa el retroceso de carro.

Al igual que en casos vistos anteriormente, las cadenas también son objetos, de tal forma que permite incorporar una serie de funciones bastante útiles, como pueden ser, entre otras:

- **capitalize()** → Devuelve una cadena con la primera letra mayúscula
- **center (num)** → Permite centrar una cadena según el número de caracteres que le pasemos por parámetro, haciendo uso de los caracteres en blanco.
- **ljust()** y **rjust()** → Para justificar tanto a izquierda como a derecha.
- **count (subcadena)** → Devuelve el número de veces que aparece la cadena pasada por parámetro.
- **find (subcadena)** → Devuelve la primera posición en la que aparece la cadena pasara por parámetro.
- **upper()** → Para convertir a mayúsculas.
- **strip ()** → Para eliminar los espacios en blanco.
- **split (carácter)** → Permite dividir una cadena en distintas partes según el separador que pasemos por parámetro.
- **splitlines()** → Permite dividir en distintas líneas.
- **len (cadena)** → Devuelve la longitud de un determinado objeto.
- **join (cadena)** → Permite unificar dos cadenas.
- **format (valores)** → Permite formatear la cadena de entrada en función de los valores pasados por parámetro.

#### ○ Diccionario o tablas hash

Hacen referencia a un conjunto de estructuras que almacenan información mediante clave en vez de por posición. Python los incluye como tipo base y no son muy complejas de utilizar.

A la hora de definir un diccionario, podemos realizarlo mediante llaves (**{}**), separando los registros por coma (,) y la clave de su correspondiente valor, mediante los dos puntos (:).

Una vez creado, podemos tener acceso a cada uno de sus elementos de la siguiente forma:

- **dval["clve"]** → donde el valor que se puede utilizar dentro va a ser el nombre o el valor de la clave que utilicemos. No puede ser su posición.

Las claves que más se utilizan son de tipos básicos numéricos o bien cadenas.

En caso de que sea necesario, tenemos la opción de poder extender el diccionario en cualquier momento, simplemente utilizando la notación de los corchetes y una clave que no hayamos utilizado hasta el momento.

Para **eliminar un registro**, simplemente utilizaremos la función **del** junto con el registro que deseemos borrar, (del dVal["clave"]). También podemos utilizar la función **clear()** para el vaciado completo.

#### ○ Listas

Las listas utilizadas en Python se refieren a las estructuras dinámicas, que permiten modificar su número de elementos. Las podemos definir de la siguiente forma:

```
Lista1 = ["valor1", 2, 2+4i]
```

Cuando necesitemos realizar alguna modificación sobre la tabla, comenzaremos siempre por el valor de índice en cero.

En Python, los índices correspondientes a las listas pueden ser positivos o negativos si se empieza desde el final. Donde Lista1[-1] correspondería a la última, Lista1[-2] a la penúltima y así sucesivamente.

Conseguir recorrer la lista por sus elementos de forma consecutiva, no es complicado, basta con indicar tanto el índice inicial como el final y separarlos por los dos puntos (:).

Podemos distinguir dos operadores diferentes para las listas, de la misma forma que en los diccionarios:

- Operador concatenación (+)
- Operador de repetición (\*)
- Para aumentar los componentes de una lista:
  - **append(valor)**→ añade el valor pasado por parámetro a la tabla.
  - **insert(posición , valor)**→ inserta el valor pasado por parámetro a la posición que se le indica también, desplazando los demás datos.
  - **extends(iterable)**→ concatena la lista actual a la que se le pasa por parámetro, originando una nueva lista con todos los valores.
- Para disminuir los componentes de una lista:
  - **pop(valor)**→ devuelve el último elemento de la lista.



- **remove(valor)**→ Elimina el primer elemento que encuentre y tenga el mismo valor que la variable que se le pasa por parámetro. En caso de que no exista, devolverá un error.

Es importante el orden que puede tener una determinada lista, desde su posición inicial hasta la final. Existen una serie de mecanismos que nos ayudan a buscar determinados elementos en una lista, como pueden ser, entre otros:

- **index(valor)**→ Devuelve la posición en la que se encuentra en la lista el valor que pasamos por parámetro.
- **in**→ Devuelve si algún elemento se encuentra en laguna posición, como, por ejemplo, ("a" in Lista1).

Para finalizar, indicaremos una función bastante importante en las listas, como es el mapeo, que consigue aplicar una determinada función a cada elemento de la lista, bajo la siguiente sentencia:

**[operación for variable in lista condición\_opcional]**

### ○ Tuplas

Las tuplas son muy parecidas a las tablas, tanto en el tamaño como en su contenido. Aunque, en el caso de las tuplas, van a actuar como listas que no se pueden modificar, por lo que son bastante más ligeras que las listas e incluso más eficientes.

Para definir las tuplas, lo podemos hacer de la siguiente forma:

**(Tupla1 = ("valor1", 2, 2+4j))**

Para poder acceder a algún elemento de la tupla, lo haremos de la misma forma que en las listas:

**Tupla1[2]**

Aunque no es posible realizar ninguna asignación como Tupla1[2]=3.

Otra característica importante de las tuplas que no la tienen las listas es que las tuplas cuentan con la posibilidad de utilizar los diferentes índices para poder seleccionar algún elemento determinado o incluso un rango de valores.

Como las tuplas son objetos fijos, no cuentan con métodos que le permitan modificar el tamaño de las mismas, salvo el método **index(valor)** que devuelve la posición del valor que corresponda con el que se pasa por parámetro.

Las listas y las tuplas, al ser tan parecidas en su funcionamiento, tienen la posibilidad de poder crear una a partir de otra.

### ○ Variables

**Las variables en Python, al igual que en JavaScript no necesitan ser definidas para su utilización.** Cuando necesitemos hacer uso de una nueva variable, sólo le tenemos que asignar el valor que deseemos y, a partir de ese momento, ya está accesible.

Podemos diferenciar **dos tipos de variables** dependiendo del ámbito en el que vayan a estar visibles:

- **Globales:** van a tener un determinado valor a lo largo de todo el programa, para las distintas funciones y clases que definamos.
- **Locales:** sólo van a tener sentido dentro del bloque en el que se inicialicen, como, por ejemplo, dentro de una determinada función.

Como ya indicamos anteriormente, no es posible realizar la asignación en línea tanto para lenguajes tipo C o Java, pero sí podemos hacer algo parecido mediante las tuplas. Las tuplas sí se pueden asignar entre ellas, de tal forma que, podemos tener en una parte una asignación de un número de elementos y, en la otra, otra asignación diferente, obteniendo como resultado que cada elemento, se asigne con un valor correspondiente del otro lado.

Para definir las tuplas, lo podemos hacer de la siguiente forma:

**(Tupla1 = ("valor1", 2, 2+4j))**

A modo de ejemplo, quedaría de la siguiente forma:

**((x, y, z)=(1, "ABC", True)),** obteniendo, por tanto que:

**x vale 1, y vale "ABC", z vale True**

### ○ Impresión por pantalla

Para imprimir por pantalla, vamos a utilizar una variable denominada cadena de formato. La función que utilizaremos para mostrar por pantalla va a ser la función `print` a la que le pasamos una cadena que va a determinar la forma en la que deseamos imprimir las variables, especificando su posición, tipo y formato de cada una, con unos caracteres de formato y las distintas variables a utilizar.

La función ***print***, dentro de la impresión, tiene una serie de características, como pueden ser que:

- Si separamos los valores por comas (,), se va a imprimir, de forma automática un espacio entre ellos.
- Cuando utilicemos el operador de concatenación (+), tendremos que añadir nosotros los espacios en el lugar necesario
- Debemos convertir a cadena, mediante la función **`str()`**, aquellas variables que no sean de este tipo.

### ○ Control de flujo

Como hemos indicado en anteriores apartados, no disponemos de ningún elemento para poder indicar la finalización de un bloque, por lo que estableceremos el contenido del mismo mediante el sangrado, indicando el mismo número de espacios a todas las líneas que formen parte de él.

De esta forma, tendremos distintos niveles de sangrado dependiendo de la cantidad de bloques existentes.

- **Condicionales.** Vamos a comenzar con los condicionales, detallando su construcción y funcionamiento.

```
if num=1:                                #comprobamos los dos puntos e iniciamos
el bloque
    print "uno"                          #sangrado que diferencia la parte que cumple
la condición
else:                                    #para los demás casos, sangrado igual que
el if
    print "Distinto de uno"              #sangrado para la parte falsa
```

```

if num=1:                                #No hay estructuras switch case
    print "uno"
elif num==0:
    print "Cero"
else:
    print "Distinto de uno y de cero"

```

Mediante la cláusula `if`, podemos realizar un conjunto de sentencias en función de una determinada condición.

- La variante más simple que se presenta no cuenta con la parte del `else` y va a ejecutar todas las líneas que cumplan una condición.
- Otra de las formas en las que puede aparecer es con las dos partes (`if-else`), ejecutando la parte del `if` en caso de que la condición sea verdadera y, para los demás casos, ejecutará la parte del `else`.
- Otro caso que debemos señalar es que este lenguaje no cuenta con la tradicional sentencia `switch` o `case`. Por tanto, incorpora una estructura
  - `elif <condicion>`
  - Que podemos añadir a la condición `if` y `else` diferenciando los casos que sean necesarios.

Debemos señalar también que, al igual que muchos otros lenguajes, Python también puede utilizar estructuras parecidas al operador condicional (`?:`). Veamos un ejemplo en el que se debe mostrar por pantalla "cero", si la variable tiene ese valor o "Distinto de cero" si tiene cualquier otro valor.

```
print "cero" if (valor==0) else "Distinto de cero"
```

### ○ Bucle for

Cuando tenemos que recorrer un objeto repetitivo (listas, tuplas, etc.) podemos hacer uso de la estructura for, cuya sintaxis es la siguiente:

#### for varInicio in Objrepetitivo

El bucle debe recorrer cada uno de los componentes que pertenezcan a la lista, tomando un valor correspondiente en cada iteración.

Por ejemplo, si disponemos de cinco valores, vamos a mostrar aquellos que sean pares.

```
print "-----bucle for-----"
tvalor=(1,2,3,4,5)
for ele in tvalor:
    if ele%2==0: print str(ele) + "el elemento es par"
for ele in range(5,25):
    print "El valor es: ",ele
```

### ○ Bucle while

Python sólo puede utilizar el bucle *while* <condición>. No puede hacer uso de la otra opción de la estructura *do- while*.

Bucle 1	Bucle 2
<pre>valor1=0; while valor&lt;20:     print "Valor: ", valor1     valor1+=2 print "FIN"</pre>	<pre>valor2="" while True:     valor2=raw_input("ADIOS&gt;")     if valor2 == "adios":         break print valor2</pre>

El primer bucle va a estar repitiéndose hasta que **valor1** tome el valor de **15**. Una vez que su valor sea 20, ya no se va a repetir más. En cada iteración que se produzca, va a ir aumentando la variable de dos en dos, mostrando sólo aquellos valores que sean pares.

Y el segundo bloque, va a utilizar la palabra **break** para salir del bucle una vez que el usuario escriba la palabra **ADIOS**.

### ○ Programación Orientada a Objetos

El lenguaje Python, aparte de ser un lenguaje estructurado, también es un lenguaje orientado a objetos, por lo que utiliza todas las ventajas de este tipo de programación.

A la hora de definir una clase, utiliza la siguiente sintaxis:

**class** nombre\_clase:

Dentro de la clase, debemos poner la información con su correspondiente sangrado.

Veamos el siguiente ejemplo en el que declaramos una clase **aula**, con dos métodos principales, que van a ser: **getTipo()** y **setTipo()** que vamos a utilizar para establecer los correspondientes tipos declarados en el constructor.

```
class aula:                                #primer parámetro siempre self en
todos
    def __init__(self, tipo):               #constructor de la clase
        self.tipo=tipo                     #propiedad pública
    def getTipo(self):                      #método getter
        return self.tipo
    def setTipo(self, tipo):                #método setter
        self.tipo=tipo
```

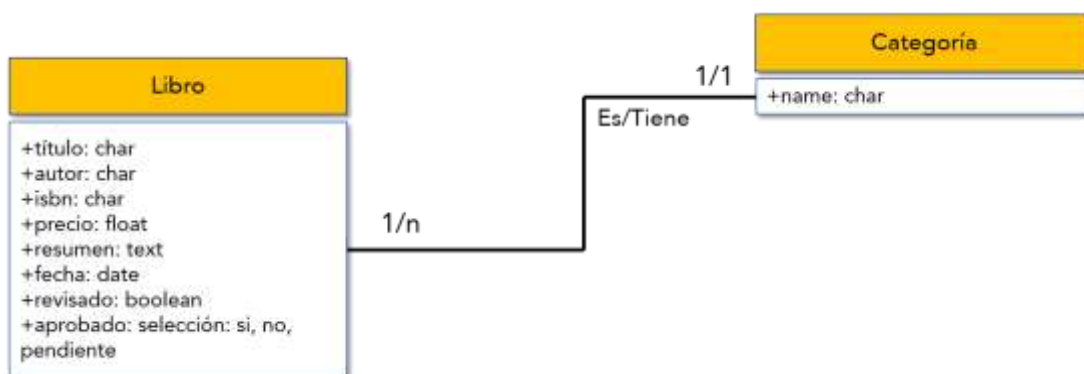
Disponemos de una serie de métodos ya implementados que podemos utilizar en las clases en cualquier momento de nuestro programa. A continuación, vamos a indicar algunos de ellos, como pueden ser:

- **\_init\_(self, args)** Constructor
- **\_del\_(self)** Destructor
- **\_deltiem\_, \_getitem\_, \_setitem\_** Se van a implementar cuando queramos ofrecer a la clase de la misma funcionalidad del diccionario.
- **\_str\_(self)** Para realizar una representación en cadena de la clase.
- **\_cmp\_(self, otro)** Compara dos cadenas, devolviendo:
  - Si el primer elemento es menor, el valor de -1.
  - Si son iguales, devuelve 0.
  - Si el segundo elemento es menor, devuelve 1.
- **\_len\_(self)** Utilizado con la sentencia **len(ObjetoClase)**. Va a determinar la longitud de la clase.

En la POO, podemos determinar si dos objetos son iguales mediante los operadores:

- **is** → Determina si dos objetos son iguales o no en la memoria.
- **igualdad (==)** → Va a establecer cuando dos objetos son iguales (mismo valor). Para ello, utiliza el método **\_cmp\_** ya implementado en la clase.
- **Objetos, campos y métodos**

Como ya venimos indicando en el desarrollo de esta Unidad Formativa, en OpenERP, todo son objetos. Cada tipo de recurso puede tener un objeto diferente con todos sus datos correspondientes para poder acceder a todos los elementos del tipo.



Podemos transformar todas las tablas en objetos mediante la utilización de un mecanismo de **ORM** (*Mapeado de objetos relacionales*) que puede hacer de puente entre la estructura física de la base de datos y los objetos del sistema.

Los objetos del sistema se pueden modelar mediante la definición estática de Python para el comportamiento y sus datos, además de una definición SQL del almacenamiento en la correspondiente Base de Datos.

Cuando creamos un objeto nuevo, es conveniente que implementemos una clase que herede del objeto **osv.osv**. De esta forma, podemos añadir una serie de elementos para que reconozcan el objeto nuevo.

Al definir un objeto, podemos utilizar un conjunto de propiedades ya predefinidas que comienzan siempre por el carácter de subrayado, entre las que distinguimos las siguientes:

- **\_auto**. Su valor por defecto es True, y nos permite especificar si la tabla se va a crear de forma automática o no.
- **\_columns**. Hace referencia a un diccionario de datos obligatorio que va a determinar los campos de la tabla a crear.
- **\_constraints**. Va a determinar las diferentes restricciones de los campos.
- **\_sql\_constraints**. Va a determinar las diferentes restricciones de los campos a nivel SQL.
- **\_defaults**. Corresponde a los valores de los campos por defecto.
- **\_inherit**. Objeto principal del que va a heredar el que definamos.
- **\_inherits**. Diccionario que dispone de una serie de objetos de los que a heredar el que estamos definiendo. No puede ser utilizado a la misma vez que el anterior.
- **\_log-access**. Cuando vale True, permite añadir a la tabla cuatro campos diferentes que determinan quién, cuándo se creó el registro y quién y cuándo ha sido modificado por última vez.
- **\_name**. Indica el nombre del objeto y es un valor obligatorio.
- **\_order**. Orden de los diferentes campos que vamos a utilizar en las operaciones de lectura y búsqueda.
- **\_rec-name**. Nombre correspondiente al campo que podemos utilizar en las operaciones de búsqueda. Por defecto, su valor es name.
- **\_sequence**. Nombre de la secuencia SQL que puede utilizar la tabla a la hora de crear los distintos identificadores.



- **\_table.** Nombre de la tabla que creemos.
- **Herencia**

Cualquier sistema orientado a objetos va a hacer uso de la herencia. En este apartado, vamos a definir la herencia basándonos en OpenERP y en los objetos correspondientes del sistema. Por tanto, podemos diferenciar entre tres tipos diferentes de herencia que se van a basar en los valores de las propiedades `_name`, `_inherit` e `_inherits`.

- **Herencia por extensión**

Permite añadir atributos a un objeto que ya existe, por tanto, no necesita crear ninguno nuevo. Lo vamos a utilizar en el caso en el que las propiedades `_name` e `_inherit` sean iguales.

Si deseamos que los campos que se creen nuevos se puedan ver en las vistas, debemos extender estas vistas del objeto del que heredamos y, a continuación, añadirlos de la misma forma.

- **Herencia por prototipo**

En este caso, se va a crear un objeto nuevo que va a heredar todos los componentes del padre, aunque va a ser independiente.

Crearemos la herencia cuando la propiedad `_name` tenga un valor no registrado y, la propiedad `_inherit` tenga un valor registrado. De esta forma, se van a ir añadiendo al objeto nuevo, todos los campos que estén definidos en el heredado.

- **Herencia por delegación**

Utilizaremos este tipo de herencia cuando sea necesario crearnos un nuevo objeto a partir de otros ya existentes (herencia múltiple).

Para ello, tendremos que establecer un nombre de objeto que no esté registrado en la propiedad `_name` y crear el diccionario correspondiente con todos los objetos existentes que vayan a heredar, con la variable `_inherits`.

Por tanto, el nuevo objeto va a tener todos los campos de todos los objetos más todos los que se le añadan.

### ○ Definición de campos

Cada objeto debe determinar cuáles son los datos que lo componen. Estos datos son los que tenemos que crear en la base de datos, teniendo en cuenta que, podemos heredar de otros objetos. Para tal fin, debemos hacer uso de la propiedad **\_columns** en la que podemos definir un diccionario con los valores correspondientes.

La clave de este diccionario, debe ser el nombre del campo que tenemos que crear junto con el valor de la definición.

A continuación, vamos a ver un listado con algunos de los campos que podemos utilizar:

- **boolean**. Puede tomar los valores de verdadero o falso.
- **integer**. Cualquier valor entero.
- **float**. Valor decimal. Mediante la opción **digits=(A,B)** permite que **A**, indique la cantidad de cifras enteras y, **B**, indique el número de dígitos decimales.
- **char**. Cadena de texto de tamaño determinado, mediante **size=valor** indica el valor de la cadena.
- **text**. Cadena de texto que no tiene longitud.
- **date**. Hace referencia a una fecha concreta.
- **datetime**. Fecha y hora.
- **binary**. Campo binario.
- **selection**. Permite crear una lista con la cantidad de valores que puede tomar.

Una vez que seleccionemos el tipo, es conveniente establecer una descripción del campo, menos en el caso de **selection**, ya que la descripción va situada después de la definición de valores.

Algunos de los diferentes tipos que podemos añadir a nuestras propias funciones, pueden ser:

- **required=True**. Cuando el campo es obligatorio.
- **select=True**. Permite crear un índice sobre el campo.
- **translate=True**. Indica al sistema que puede mostrar traducido el campo.
- **ondelete="set null|cascade|set default|restrict|no action"**. Sólo se puede utilizar para las restricciones, indicando cómo debe ser su comportamiento cuando se elimine el padre.

- **readonly=True.** Determina que el campo sea de solo lectura.

Si bien es cierto que los tipos básicos construyen una parte imprescindible para conseguir almacenar la información de forma correcta, no llegan a conseguir representar un esquema relacional, ya que, para ello, necesitan de un poco más de información, como son las distintas relaciones entre los objetos.

En OpenERP, podemos diferenciar cuatro tipos de relaciones que se pueden crear, como son las relaciones uno a uno, uno a muchos, muchos a uno y muchos a muchos.

Podemos definir las relaciones haciendo uso del tipo **many2one**, donde, en el primer parámetro especificaremos el objeto en el que pretendemos crear la relación y, en el segundo, la descripción del campo.

### 3.2. Entornos de desarrollo y herramientas de desarrollo en sistemas ERP y CRM.

- **Instalación de Python**

Para programar en Python, lo único que necesitamos es tener instalada una versión del intérprete junto con un editor de texto simple.

Aunque la configuración básica no es la más adecuada, nosotros lo vamos a desarrollar bajo Eclipse complementado para el desarrollo bajo Python, ya que, de esta forma, nos va a permitir contar con todas las facilidades más actuales de programación, sangrado de líneas, resaltado, autorrelleno, etc.

- **Instalación del intérprete Python**

Podemos descargar la versión deseada desde la página oficial si lo vamos a hacer desde Windows

<http://www.python.org/getit/>

Si es desde Linux, no es necesario ya que la mayoría de las distribuciones ya viene con los paquetes necesarios incluidos para su funcionamiento.

- **Descarga e instalación de Eclipse**

Descargamos Eclipse

<http://eclipse.org/downloads/>

Y, a continuación, podemos instalarlo en nuestro ordenador.

- **Instalación de PyDev para Eclipse**

Este módulo, PyDev, está desarrollado para Aptana que nos va a permitir utilizarlo dentro de Eclipse para programar bajo Python.

Desde **Help**, seleccionamos **Install New Software** lanzando el asistente que incorpora el asistente en eclipse y nos va a permitir que podemos añadir distintos módulos.

- **Configuración de PyDev para Eclipse**

En las preferencias de eclipse, si observamos el árbol que muestra las distintas categorías, desplegamos el elemento PyDev, buscando la subcategoría **Interpreter Python**.

En esta opción, vamos a poder configurar la ruta del intérprete que instalamos en el primer paso de la instalación

En la mayoría de los casos, si pulsamos el botón **Auto Config**, debe detectar los valores necesarios de forma correcta. Si no fuera así, debemos añadirlos a mano.

- **Primer proyecto**

Vamos a crearnos nuestro primer proyecto para comprobar que todo funciona de forma correcta. En este caso, vamos a desarrollar un ejemplo muy frecuente en referencia a la programación, como es el "Hola Mundo".

Veamos, paso a paso el desarrollo de nuestro ejemplo.

- Lo primero que haremos va a ser **crear el proyecto Python**, mediante **File/New/Pydev project**.

Comprobaremos que el nombre del proyecto es el correcto, seleccionamos Python en tipo de proyecto junto con la versión

correspondiente que estemos utilizando. También seleccionamos la opción de crear un subdirectorio por defecto para las fuentes **src**.

- Si abrimos el directorio **src**, podemos añadir un módulo, seleccionando la opción **New/PyDev Module**. En la ventana que se muestra, tenemos la opción de incluir los valores de **Name**, que en nuestro ejemplo se corresponde con **HolaMundo**. Y en el caso **Template** seleccionamos **<Empty>**.
- En caso de que nos encontremos trabajando bajo un sistema operativo de Windows, es habitual que el cortafuegos lance una excepción en la que tendremos que confirmar para desbloquear eclipse. Debemos aceptar la petición.

En caso de trabajar con Linux, debemos asegurarnos que el cortafuegos no filtre los puertos correspondientes de eclipse.

- Llegados a este punto, ya podemos crear nuestro código fuente. Desplegamos el archivo *holaMundo.py* y escribimos:  

```
print "Hola Mundo"
```

Podemos guardar y ejecutar nuestro fichero. Al pedirnos el tipo de ejecución que va a lanzar, seleccionamos **Python run** y, en la parte inferior de la pantalla, veremos la ejecución.

### 3.3. Operaciones con Datos en los objetos

Al tratarse de un motor de bases de datos, es una tarea normal el declarar las tablas donde se van a almacenar los datos y a continuación realizar una carga inicial. Este sistema al estar relacionado con distintos módulos, uno de los mejores lenguajes para la importación de datos es el lenguaje de marcado XML.

Necesitamos seguir el siguiente formato: `___openerp___.py`

Este fichero tendrá el siguiente formato:

```
<?xml versión="1.0"?>
```

```
<terp>
```

```
<data>
```

```
<record>
```

```
/*En el interior de esta marca se indicará la información*/
```

```
</record>
```

```
</data>

</terp>
```

Cada etiqueta `<record>` indicará un registro a insertar, debemos de indicar todos los campos requerido de la tabla.

### 3.4. Extracciones de informaciones contenidas en sistemas ERP-CRM, procesamiento de datos.

Debido a la importancia que tiene los datos en estos tipos de herramientas, la aplicación ERP- CRM favorece la exportación e importación de los datos. En el apartado anterior hemos podido comprobar cómo se hace posible la carga inicial en las tablas mediante archivos XML.

Pero además esta herramienta también cuenta con la manipulación de los datos por archivos .CSV.

De esta forma estandariza la migración de datos a otros entornos y viceversa ya que estos ficheros se pueden crear desde cualquier aplicación de hojas de cálculos o editor de texto haciendo uso de un carácter delimitador.

Las herramientas existentes en el mercado facilitan mediante interfaz gráficas la operación de la migración de datos. De esta forma cualquier usuario podrá realizar la importación o exportación de los mismo, aunque en la mayoría de las ocasiones serán los técnicos, los encargados de realizarlas.

### 3.5. Llamadas a funciones, librerías de funciones (API).227

Es conveniente, siempre que trabajemos con lenguajes estructurados, la utilización de funciones para conseguir un correcto funcionamiento. Por eso, con Python sucede de la misma forma, cuando necesitamos crear alguna función en cualquier parte del código, utilizaremos la palabra clave **def**, seguida del nombre que elijamos para la función y, entre paréntesis, indicaremos aquellos parámetros que necesitamos (entre creo y los que necesitamos).

Una vez que tengamos definida la función, podemos utilizarla cada vez que lo necesitamos, de la siguiente forma:

- **Nombre de la función** (valores que podemos utilizar)

Una de las ventajas que presenta Python con respecto a los demás es que, permite pasar como parámetros, cualquier tipo de datos, y la función tiene que ser la que se encargue de realizar las conversiones correspondientes. La

función, como tal. Debe devolver un valor a través de **return**. En caso de que se nos pase, automáticamente devolverá **None**.

En algunas ocasiones, podemos necesitar hacer una llamada a una función, aunque con menos parámetros de los que habíamos definido en un primer momento. En este caso, añadiremos una asignación, a continuación de la definición del parámetro, indicándole al sistema que, si ese valor no está establecido, haga uso del que nosotros le estamos indicando. Esta característica es conocida con el nombre de parámetros optativos o parámetros con valores por defecto.

En cuanto a las funciones, podemos señalar una característica muy importante, como puede ser, la posibilidad que existe de que podamos reordenar los parámetros cuando se realice una llamada a una determinada función, simplemente anteponiendo el nombre del parámetro que hemos definido a otro valor, mediante una asignación y así podremos situarlo en el lugar que deseemos.

Igual que el lenguaje de programación JAVA, existe un repositorio de funciones llamados API, que, de forma oficial, nos ofrece una serie de colecciones y métodos para facilitarnos la implementación de los códigos fuentes.

Para acceder a dicha API podemos hacer clic en  
<https://docs.python.org/3/library/index.html>

### 3.6. Depuración de un programa.

Una vez que hemos desarrollado un código determinado en Python, para asegurarnos de que es correcto, podemos hacer uso de un IDE o algún editor que reconozca este lenguaje para conseguir eliminar aquellos errores sintácticos que se detecten en el código.

Para ficheros XML es conveniente que los testeemos mediante un parser XML o haciendo uso de un determinado navegador de Internet para comprobar que la estructura es la adecuada. En caso de que existan etiquetas mal balanceadas, que falten etiquetas o algunos errores típicos, se descubrirán de una forma bastante rápida.

Una vez que ya hemos detectado los posibles errores y solventado los fallos, ya podemos pasar a la depuración del código ejecutable. En este caso, vamos a utilizar el depurador de Python integrado. Para ello, debemos importar en nuestro código el paquete `pdb` para poder establecer la sentencia `pdb.set_trace()` al principio del módulo a depurar.

Ya configurado el código, lanzaremos el servidor mediante la opción debug, forzando de esta forma al sistema OpenERP a abrir una consola de depuración cuando llegue a nuestro módulo. Cuando se abra la consola, disponemos de una serie de comandos para poder inspeccionar los diferentes valores, como pueden ser, entre otros, establecer puntos de ruptura y ejecutar paso a paso.

### 3.7. Manejo de errores

- Control de errores

A lo largo de un programa pueden ir apareciendo errores por distintas situaciones:

- Código que no está bien programado
- Datos mal introducidos
- Otras causas

Independientemente de la causa que sea, es fundamental que desarrollemos una gestión adecuada del fallo producido para evitar que la aplicación termine sin ningún tipo de explicación. Por tanto, para conseguir una gestión adecuada sobre los errores, el sistema nos debe permitir presentar un modelo que esté basado en la gestión de las Excepciones correspondientes.

**Una Excepción no es más que un objeto que tiene información acerca de un error que ha sido producido sobre el proceso de ejecución**

Todas las excepciones que se generen deben acogerse para ser tratadas, si no, el intérprete puede proporcionar un mecanismo asignado por defecto que muestre un mensaje por pantalla y consiga frenar la aplicación.

Las Excepciones son objetos explicativos, que van a tener asignado un tipo definido en el lenguaje proporcionando una serie de datos que detallen las causas que han generado el fallo. Por tanto, una Excepción `IOError` va a determinar un error en una operación de entrada- salida, una de `MemoryError` es la que se va a crear cuando no quede suficiente memoria, etc.



Las excepciones son lanzadas por el sistema y debemos capturarlas para evitar un determinado comportamiento. Podemos gestionarla mediante las construcciones **try...except...else...finally** (try catch de Java)

Dentro del bloque **try** podemos capturar un error que puede venir determinado con distintas sentencias **except**.

En el caso en el que vayamos a tratar dos errores de la misma forma, podemos incluir una línea **except** a ambos, encerrados entre paréntesis y separados por comas.

Podemos incluir un bloque **else** para ofrecer un mayor tratamiento en caso de que se produzca algún error

Y, dentro de **finally** podemos situar las sentencias que se van a ejecutar una vez que se trate un determinado problema.

Como ya hemos indicado, el sistema es el que va a generar las excepciones, aunque algunas veces, es conveniente que nosotros mismos hagamos uso de algún mecanismo para informar del error. En cualquier momento podemos lanzar una excepción utilizando la palabra **raise** junto con algún objeto que herede de **Exception**.

A continuación, vamos a señalar algunas de las excepciones más conocidas:

- **BaseException**. Permite crear una clase base para todas las demás.
- **Exception**. Clase base para todas aquellas excepciones que no se correspondan con entrada- salida.
- **AithmeticError**. Clase base para aquellos errores aritméticos, pudiendo extender de ella los siguientes: `FloatingPointError`, `OverflowError`, `ZeroDivisionError`, etc.
- **EOFError**. Indica que hemos sobrepasado el fichero.
- **IOError**. Indica que se ha producido un error de entrada- salida.
- **ImportError**. Indica error en la importación. Además, nos permite comprobar las diferentes funcionalidades del sistema.
- **IndexError**. Error en el índice de acceso.
- **KeyError**. Cuando nos encontramos con una clave que no existe.

- **Documentación**

Realizar la documentación de un código siempre es una labor muy importante, ya que permite facilitar la tarea tanto a nosotros, como a futuros programadores que tengan que trabajar con el mismo código. Además, Python con algunas facilidades que podemos utilizar para la documentación.

Primero, podemos hacer uso de la almohadilla (#) para comentar las líneas que deseemos hasta el final de línea. También lo podemos hacer utilizando las triples comillas simples que nos van a permitir comentarios en varias líneas.

También nos podemos encontrar con el caso en el que la persona que actúe como intérprete haga uso también de nuestras anotaciones. Entonces, es conveniente que documentemos nuestra función o clase mediante una cadena que explique el funcionamiento y los componentes principales.

Podemos escribir esta cadena en la línea siguiente a la de la definición de una determinada función (**def Mifunión (...)**), cuando creamos la clase (**class Miclase...**), o al comienzo de algún módulo, utilizando triples comillas, después de la codificación de caracteres.

Si bien es cierto que el contenido que podemos incluir es libre, es conveniente utilizar las recomendaciones de *javadoc*, como pueden ser:

- @param, para indicar un parámetro.
- @return, que indica el valor devuelto.
- @author, nombre del autor.
- @version, versión a la que corresponde, etc.

## Bibliografía

César San Juan Pastor, "Sistemas de Gestión Empresarial". Garceta, 2015.

Webgrafía

<http://www.dataprix.com>

```
function updatePhotoDescription() {  
    if (descriptions.length > (page * 9) + (currentImage subding() - 1)) {  
        document.getElementById('bigImageDesc').innerHTML = descriptions[page * 9 + (currentImage subding() - 1)]  
    }  
}  
  
function updateAllImages() {  
    var i = 1;  
    while (i < 10) {  
        var elementId = 'foto' + i;  
        var elementIdBig = 'bigImage' + i;  
        if (page * 9 + i - 1 < photos.length) {  
            document.getElementById(elementId).src = 'images/' + photos[page * 9 + i - 1] + '.jpg';  
            document.getElementById(elementIdBig).src = 'images/' + photos[page * 9 + i - 1] + '.jpg';  
        } else {  
            document.getElementById(elementId).src = 'images/default.jpg';  
            document.getElementById(elementIdBig).src = 'images/default.jpg';  
        }  
        i++;  
    }  
}
```