Fundamentos tecnológicos

Daniel Santanach Casals

PID_00203135

Universitat Oberta de Catalunya

Índice

Introducción

Actualmente, podemos encontrar en el mercado multitud de proveedores que ofrecen una gran cantidad de herramientas que permiten conformar sistemas y soluciones de gestión documental. Dentro de esta diversidad podemos identificar elementos comunes que tratamos en este módulo, desglosados en las siguientes temáticas.

En primer lugar, revisaremos cuáles son las arquitecturas más utilizadas para la implementación de sistemas de información y estudiaremos las principales características de aquellas que habitualmente se utilizan en sistemas de gestión documental.

En segundo lugar, veremos qué son y qué aportan los estándares tecnológicos a los sistemas de información, así como sus diferentes tipologías.

En tercer y cuarto lugar, estudiaremos los diferentes tipos de unos componentes muy importantes en cualquier sistema de información como son las bases de datos y los motores de búsqueda.

Finalmente, revisaremos las principales características de implementaciones concretas de varios tipos de sistemas de gestión de datos, como son los sistemas ERP, CRM, BPM, BI y ECM.

Objetivos

Con el estudio de este módulo, se pretende alcanzar los siguientes objetivos:

1. Conocer las características arquitectónicas de un sistema de información.

2. Saber qué es un estándar tecnológico.

3. Aprender las tipologías de bases de datos.

4. Aprender las tipologías de los motores de búsqueda.

5. Conocer las principales características de sistemas específicos de gestión de datos.

1. Arquitectura de sistemas

En esta unidad conoceremos el concepto de sistema de información y el concepto de arquitectura de un sistema de información para comprender qué es y cómo se integra dentro de una organización un sistema de gestión documental.

Además estudiaremos criterios con los que seleccionar una arquitectura y los distintos tipos de arquitectura: arquitectura lógica, física, patrones de arquitectura, arquitecturas centralizadas o monolíticas y arquitecturas distribuidas.

Por último, explicaremos brevemente que es el *cloud computing*, su tecnología y su relación con la gestión documental.

1.1. Conceptos

Un **sistema de información** (SI) es un conjunto de elementos (personas, datos, actividades, recursos materiales) que actúan de manera conjunta para capturar, procesar, almacenar y distribuir datos e información con una determinada finalidad. Esta finalidad suele ser la toma de decisiones, la coordinación, el control y el análisis dentro de una organización.

En la mayoría de sistemas de información los recursos materiales están constituidos por recursos informáticos y de comunicaciones, y es por ello por lo que se suele entender el concepto de "sistema de información" como "sistema de información informático". Así lo entenderemos nosotros a lo largo del texto. No obstante, esto no es estrictamente así y deberíamos considerar un sistema de información informático como un subconjunto de los sistemas de información en general.

En un sistema de información, de la misma manera que en un edificio o en una construcción, podemos identificar una determinada arquitectura que establece la estructura, el funcionamiento y la interacción entre los diferentes elementos que lo conforman. Es lo que llamamos la arquitectura del sistema de información. Todo sistema de información tiene una arquitectura.

Una definición formal de **arquitectura de un sistema de información** la podemos encontrar en el estándar ISO/IEC/IEEE 42010:2011, *Systems and software engineering-Architecture description*. Este estándar define la arquitectura como los conceptos o propiedades fundamentales del sistema dentro de su entorno, plasmados en sus componentes, sus relaciones, y en los principios de su diseño y evolución.

1.2. Criterios de selección de una arquitectura de sistemas

De la misma forma que los requisitos de arquitectura para construir una pequeña casa de campo son distintos a los requisitos para construir un rascacielos, la arquitectura del sistema de información para una pequeña organización que actúa a nivel local debería ser distinta a la arquitectura del sistema de una organización con miles de empleados repartidos por todo el mundo. Por lo tanto, no existe una única arquitectura para un sistema de información sino que existen muchas arquitecturas posibles y para cada sistema se deberá escoger la más adecuada.

En la selección de la arquitectura se deben tener en cuenta cuatro aspectos:

- **Objetivos**. La arquitectura seleccionada deberá ser aquella que mejor se adapte a los objetivos (funcionales, tecnológicos, de organización, de integración con otros sistemas, de negocio, etc.) que se persiguen. Esta arquitectura debe dar respuesta a las necesidades actuales, además de soportar los requisitos de mantenimiento y evolución futura del sistema.

- **Restricciones tecnológicas**. Aparte de los objetivos, también se deberán considerar las restricciones derivadas de las tecnologías de la información: hay arquitecturas que son más recomendables de implementar con determinadas tecnologías, y tecnologías que no están desarrolladas para implementar ciertas arquitecturas.

- **Estructuración y articulación de componentes**. La arquitectura debe estructurar y articular relaciones entre los componentes de un sistema de información.

- **Funcionalidades de la arquitectura**. Una arquitectura de sistemas debe cumplir los siguientes requisitos:
  - Permitir a todos los interesados en el sistema (usuarios, administradores, diseñadores, desarrolladores, etc.) tener una visión común del sistema que sirva como base a la comprensión mutua entre todos ellos.

  - Permitir acotar, visualizar y comprender la complejidad del sistema de información.

– Ser la base para el análisis, el diseño, la construcción y la evolución del sistema de información.

1.3. Tipos de arquitectura y componentes

La arquitectura de un sistema de información se puede analizar desde dos puntos de vista:

- Desde un punto de vista lógico, se habla de **arquitectura lógica**. Está formada por los componentes, subsistemas, programas y aplicaciones que definen las funcionalidades del sistema.

- Desde un punto de vista físico, se habla de **arquitectura física**. Está formada por las máquinas físicas y las redes o elementos de comunicaciones que dan soporte físico al sistema.

La combinación de una arquitectura lógica con una arquitectura física da lugar a una **arquitectura de sistemas de información**. Algunas de estas arquitecturas son consideradas arquitecturas "estándar" y sirven para resolver problemáticas concretas. A estas arquitecturas estándares las llamamos patrones de arquitectura y las veremos más adelante.

Además, para el desarrollo de la arquitectura lógica y la implementación de la arquitectura física de un sistema de información, será necesario utilizar un lenguaje o **lenguajes de programación** específicos para garantizar el correcto encaje de todos los componentes (lógicos y físicos) del sistema.

1.3.1. Arquitectura lógica

La **arquitectura lógica** agrupa los componentes de un sistema desde cuyo punto de vista se define la función que realizan dentro del mismo.

La arquitectura lógica de un sistema de información está formada por "capas" (*layers* en inglés). Tradicionalmente se identifican tres capas: la capa de presentación o de interfaz de usuario, la capa de negocio y la capa de datos.

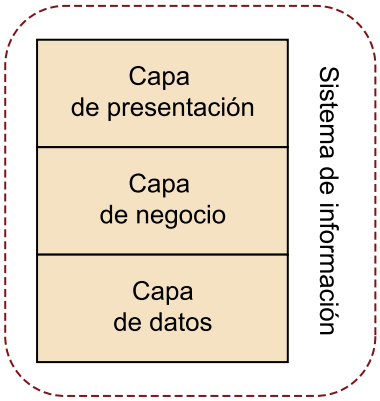

Figura 1. Arquitectura lógica de un sistema de información

La **capa de presentación** agrupa todo aquello que se enfoca a la interacción con el consumidor o usuario final del sistema. Esta capa es lo que el usuario ve del sistema y la que le permite interactuar con él. Normalmente está formada por una interfaz gráfica de usuario[1], que presenta el sistema al usuario y le permite visualizar información almacenada e introducir información nueva.

(1)En inglés, *graphical user interface* o GUI.

La **capa de negocio** es donde encontramos los componentes que gestionan los procesos y las reglas de negocio del sistema. En esta capa se encuentra la lógica del sistema, es decir, donde se ejecutan los programas que generan sus funcionalidades. Aquí es donde, ante una determinada petición con unos determinados parámetros, se decide la respuesta más adecuada, en función de las reglas del sistema (reglas de negocio). Esta capa se comunica con las otras dos capas. Con la capa de presentación se comunica para recibir las peticiones del usuario del sistema y enviarle las correspondientes respuestas, y con la capa de datos se comunica para almacenar y/o recuperar la información necesaria.

La **capa de datos** contiene los componentes dedicados a la persistencia y la gestión de los datos del sistema. Esta capa es la encargada de almacenar todos los datos del sistema, garantizar su persistencia a lo largo del tiempo y permitir tanto incorporar nuevos datos como recuperar datos almacenados.

Persistencia

La persistencia se refiere a la conservación (en este caso, de datos de un sistema) a lo largo del tiempo. Es decir, los componentes de la capa de datos garantizan que los datos de un sistema estarán disponibles en el futuro en cualquier momento.

A pesar de esta clara definición y distinción entre cada una de las capas, debemos tener en cuenta que tanto las capas de la arquitectura lógica como la separación entre ellas son elementos puramente conceptuales. Es decir, en un sistema de información, las capas lógicas no tienen por qué estar separadas unas de otras. La separación entre capas es una separación teórica y aunque en algunas arquitecturas estas capas estarán realmente diferenciadas y separadas (como veremos en las arquitecturas de n-niveles), en otras arquitecturas la frontera entre capas será difícil de identificar (como en las arquitecturas cliente-servidor) o simplemente no habrá frontera alguna (como en las arquitecturas monolíticas).

1.3.2. Arquitectura física

La **arquitectura física** de un sistema de información es la implementación en un entorno físico de su arquitectura lógica.

La arquitectura física está formada por un conjunto de componentes, como son ordenadores, impresoras, teléfonos móviles o tabletas. A estos componentes físicos los llamaremos de manera genérica "nodos". Estos componentes proporcionan soporte físico a las funcionalidades del sistema y a las relaciones que se establecen entre los componentes. Estas relaciones están basadas en redes de comunicaciones.

De la misma forma que en la arquitectura lógica hablábamos de capas, en la arquitectura física de un sistema de información hablaremos de "niveles" (*tiers* en inglés). En cada nivel de la arquitectura física podremos encontrar una o más capas de la arquitectura lógica del sistema. A su vez, cada nivel de la arquitectura física puede estar soportado por uno o varios nodos en función de las necesidades del sistema.

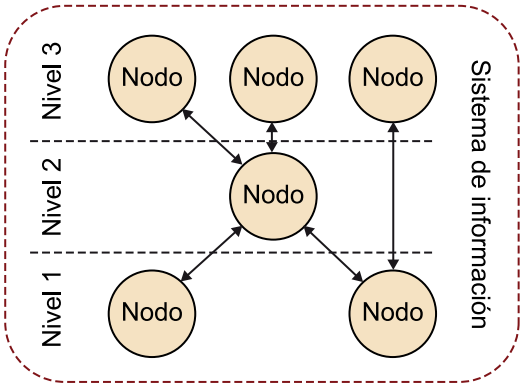

Figura 2. Arquitectura física de un sistema de información

Según donde residan las tres capas hablaremos de arquitecturas centralizadas o arquitecturas distribuidas. Podemos hacer residir las tres capas de la arquitectura lógica de un sistema en un único nodo, y por eso hablaremos de **arquitecturas centralizadas**. O las podemos repartir entre varios nodos en varios niveles, y en ese caso hablaremos de **arquitecturas distribuidas**. En una arquitectura distribuida podemos tener, por ejemplo, cada capa lógica en un nodo físico, dos capas compartiendo un mismo nodo, o una capa repartida entre varios nodos.

1.4. Patrones de arquitectura

A lo largo del tiempo y en paralelo a la evolución de los sistemas de información, se han ido definiendo lo que llamamos "patrones" de arquitectura.

Un **patrón** es una solución conocida para un determinado tipo de problema y que concentra todo el conocimiento generado a lo largo del tiempo fruto de la experiencia en un determinado dominio (en nuestro caso, el dominio de los sistemas de información).

Un sistema de información bien estructurado seguirá un determinado patrón de arquitectura. Un patrón de arquitectura para un sistema de información es una determinada combinación de capas de arquitectura lógica con niveles de arquitectura física. Cada patrón da más o menos importancia a una determinada capa o a un determinado nivel, en función de la finalidad del sistema. Por ejemplo, algunos patrones se enfocan a dar prioridad al rendimiento del sistema, esto es, que el sistema responda rápidamente a las peticiones del usuario. Otros patrones se centran en la alta disponibilidad de la información, es decir, se orientan a que el usuario pueda acceder a la información del sistema desde cualquier ubicación. Por ello, la selección de un patrón de arquitectura para un sistema de información es una de las primeras tareas que los diseñadores de sistemas deben abordar para poder dar a dicho sistema las características deseadas.

Cuando se comenzaron a desarrollar sistemas informáticos a partir de la segunda mitad del siglo XX, un reto importante era cómo organizar los diferentes elementos de un sistema de información, ante la falta de experiencia y patrones. Por aquel entonces, la definición de la arquitectura de un sistema se basaba en la intuición y las habilidades de sus diseñadores. El paso del tiempo ha permitido ir definiendo patrones de arquitecturas (tanto centralizadas como distribuidas) que han llegado a considerarse estándares o, como mínimo, se han convertido en los más comunes para resolver un determinado tipo de problemática. Cada uno de estos patrones de arquitectura tiene sus ventajas y sus inconvenientes y es necesario adoptar uno u otro en función de las características específicas del sistema de información al que deben dar soporte.

Los patrones que se han convertido en estándares son los que se explican más adelante: arquitectura monolítica, arquitectura cliente-servidor, arquitectura de *n*-niveles o arquitectura SOA.

El desarrollo de las funcionalidades incluidas en cada una de las capas lógicas del sistema de información, así como, si es necesario, la interacción entre cada una de estas capas se articula a través de los lenguajes de programación. Estos lenguajes deben también permitir la compatibilidad de las capas lógicas con los niveles físicos del sistema donde residan. Existen infinidad de lenguajes de programación que han ido evolucionando a lo largo del tiempo junto a las diferentes tecnologías y patrones de arquitectura. Algunos de estos lenguajes son genéricos mientras que otros encajan mejor o se han especializado en determinados tipos de arquitectura.

Algunos de los lenguajes son: Pascal, FORTRAN, COBOL, Basic, Delphi, Visual Basic, C, C++, Visual C, C#, Perl, PHP, Visual Basic .NET, Java.

Una lista completa se puede encontrar en: http://es.wikipedia.org/wiki/Lista_de_lenguajes_de_programación.

1.5. Arquitecturas centralizadas o monolíticas

Una **arquitectura centralizada o monolítica** es la más simple y obvia de las arquitecturas posibles. En esta arquitectura las tres capas lógicas del sistema están fuertemente entrelazadas las unas con las otras (no existe separación entre ellas) y se concentran en un único nodo (o, lo que es lo mismo, en un único nivel físico).

Los sistemas basados en una arquitectura monolítica son, normalmente, sistemas *ad hoc* (hechos a medida). Ello les confiere, por un lado, eficiencia, rapidez y seguridad porque todos sus elementos han sido diseñados e implementados para cumplir con un objetivo concreto en un entorno físico muy controlado. Pero, por otro lado, les falta flexibilidad para adaptarse a condiciones diferentes para las que han sido concebidos. Además, al ser una arquitectura tan compactada y con las capas lógicas tan fuertemente entrelazadas, presenta dificultades, por ejemplo, cuando algo falla en el sistema es difícil encontrar qué ha fallado, y cuando se necesita hacer evolucionar el sistema, resulta complicado saber dónde tocar.

Esta arquitectura era la utilizada en los primeros sistemas de información donde el sistema consistía en un único y potente ordenador central muy grande y caro[2], con un programa compuesto por varias rutinas entrelazadas que permitían ejecutar un proceso de principio a fin sin la necesidad de utilizar ninguna funcionalidad externa al propio sistema. Normalmente, el usuario podía acceder al sistema a través de terminales "estúpidos" (simples pantallas con un teclado sin ninguna capacidad de computación) conectadas directamente al ordenador central (figura 3).

(2)En inglés, un *mainframe*.

En esta arquitectura, las tres capas lógicas están mezcladas y es imposible establecer una frontera entre ellas. Usando un lenguaje más técnico, diríamos que el sistema se compone de diferentes módulos compilados y unidos entre ellos a través de un enlazador.

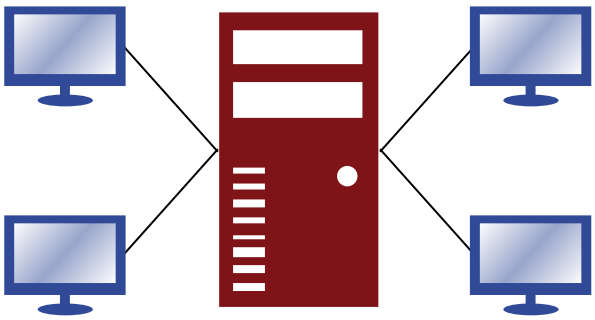

Figura 3. Arquitectura monolítica

Para poder programar los sistemas *mainframe*, se desarrollaron los primeros lenguajes de programación. Algunos de estos lenguajes son los siguientes:

- El "lenguaje máquina". Es el lenguaje del propio hardware del ordenador.
- El "lenguaje ensamblador". Es un conjunto de instrucciones "comprensibles" por el programador del sistema y que se traducían posteriormente a lenguaje máquina.
- Los lenguajes de más alto nivel, como Pascal, FORTRAN, COBOL, Basic o C, que actualmente continúan existiendo. Permiten el desarrollo de funcionalidades que son compiladas y enlazadas antes de ser ejecutadas por el sistema. Aunque inicialmente fueron concebidos para la programación de sistemas con arquitecturas centralizadas, la mayoría de estos lenguajes pueden utilizarse también para programar arquitecturas distribuidas.

Los *mainframe* empezaron a dejar de ser de uso común cuando en los años ochenta aparecieron ordenadores personales (PC) suficientemente potentes y, sobre todo, con capacidades de conexión a redes de comunicaciones que permitieron el desarrollo de arquitecturas distribuidas.

Sistemas basados en arquitectura monolíticas

Aunque no es habitual encontrar sistemas de información actuales que se basen en arquitecturas monolíticas, todavía podemos encontrar sistemas concretos que se implementan sobre este tipo de arquitectura. Un ejemplo son algunos sistemas de gestión financiera de bancos y cajas que requieren una arquitectura muy compactada donde la información esté centralizada y no se comparta (al menos directamente) con el exterior.

1.6. Arquitecturas distribuidas

Al contrario que en una arquitectura centralizada, donde un único nodo realiza todas las funciones del sistema, en una arquitectura distribuida las funcionalidades y el procesamiento de la información se distribuyen de manera independiente sobre varios nodos.

El concepto de "distribuida" no es un concepto físico sino lógico, por tanto, esta definición de arquitectura distribuida no es estrictamente cierta. Es decir, en teoría podemos tener una arquitectura distribuida sobre un único nodo físico siempre que este nodo tenga capacidad para procesar simultáneamente diferentes datos de manera independiente. No obstante, lo habitual es implementar arquitecturas distribuidas sobre varios nodos físicos, puesto que las mayores ventajas de una arquitectura distribuida se basan en el hecho de poder repartir las funciones del sistema sobre varios nodos físicos. Aunque este tipo de arquitecturas presenta ventajas, también cuenta con inconvenientes, como veremos más adelante.

En los años ochenta, las arquitecturas distribuidas empezaron a ser viables cuando en el mercado aparecieron los ordenadores personales con capacidad de proceso independiente de un ordenador central y con capacidad para conectarse a una red de comunicaciones e intercambiar información entre ellos. La bajada de los precios y la popularización de los PC permitieron pensar en utilizar su potencia de cálculo y su capacidad de almacenamiento para ejecutar procesos independientemente del ordenador central de un sistema de información. De esta manera aparecieron los terminales "inteligentes" en contra-

posición a los terminales "estúpidos" de los sistemas centralizados. Además, el precio de los ordenadores centrales de las arquitecturas centralizadas también bajó considerablemente de manera que en un sistema se podía disponer de varios de ellos y pasaron a llamarse "servidores". La combinación de las capacidades de los terminales inteligentes (también llamados "clientes") con las capacidades de los servidores dio lugar a las arquitecturas distribuidas.

En este apartado describimos las ventajas y los inconvenientes de una arquitectura distribuida, y algunos de los distintos patrones de arquitectura distribuida que podemos encontrar.

1.6.1. Ventajas e inconvenientes

Una arquitectura distribuida presenta varias **ventajas** respecto de una arquitectura centralizada:

- Permite compartir recursos entre sus diferentes componentes, ya que una máquina cliente puede usar recursos de una máquina servidor y viceversa.

- Permite la concurrencia de procesos, es decir, permite que varios procesos puedan ejecutarse a la vez en máquinas distintas.

- Es escalable, ya que se pueden añadir más maquinas, ya sean cliente o servidor, al sistema para incrementar su capacidad. No obstante, ello también puede ser contraproducente, ya que un mayor número de máquinas en el sistema puede repercutir negativamente en su rendimiento.

- Es abierta, ya que la necesidad de que las distintas máquinas puedan relacionarse, y entenderse unas con otras, obliga al uso de protocolos de comunicaciones estándar.

- Aporta una mayor tolerancia a posibles fallos en el sistema, ya que existe la posibilidad de distribuir su información y poder recuperarla íntegramente aunque exista algún problema en alguno de los componentes del sistema.

No obstante, no todo son ventajas en un sistema con una arquitectura distribuida. Algunos de los **inconvenientes** a tener en cuenta respecto de una arquitectura centralizada son:

- Una mayor complejidad por el mayor número de componentes que contiene el sistema y, sobre todo, por la necesidad de intercomunicar todos estos componentes. Por ejemplo, el rendimiento del sistema ya no depende únicamente de la capacidad de proceso de una única máquina, sino de la velocidad de la red de comunicaciones.

- Unos mayores requerimientos de seguridad, ya que el acceso a la información es posible desde varios puntos diferentes.

- Unos mayores requerimientos de administración del sistema al existir un mayor número de elementos que se deben mantener alineados (en cuanto a versiones de sistema operativo, por ejemplo) y evitar que un problema en un elemento se propague a los demás.

- Una menor predictibilidad de la respuesta del sistema, ya que el número de casuísticas a gestionar (por ejemplo, número de usuarios accediendo al sistema desde diferentes puntos, carga de la red por transferencia de datos de un punto a otro del sistema, etc.) puede impactar en el tiempo de respuesta del sistema ante una petición del usuario.

Con la evolución del hardware y de los lenguajes de programación, las ventajas de una arquitectura distribuida superan claramente a sus desventajas. Por tanto, estas arquitecturas distribuidas son las arquitecturas que encontramos en la mayoría de sistemas de información de hoy en día.

1.6.2. Patrones de arquitectura distribuida

Existen múltiples patrones de arquitectura de sistemas distribuidos, como por ejemplo, la arquitectura multiprocesador, arquitectura entre pares, arquitectura cliente-servidor, la arquitectura de n-niveles o la arquitectura SOA (*service oriented architecture*).

Estos tres últimos patrones mencionados son los más habituales de arquitectura en sistemas de información distribuidos en general, y de mayor aplicación en sistemas de gestión documental en particular. A continuación veremos estos patrones.

Arquitectura multiprocesador

La **arquitectura multiprocesador** o la **arquitectura de objetos distribuidos** es muy común en sistemas de gestión de la información en tiempo real, que requieren un alto rendimiento y una gran adaptabilidad a los cambios de su entorno.

Para obtener este rendimiento, las tareas del sistema se pueden distribuir, o bien entre varios procesadores que trabajan en paralelo (arquitectura multiprocesador) o bien entre varios objetos desarrollados para trabajar conjuntamente y distribuidos entre los varios nodos del sistema. Estos nodos se comunican entre ellos a través de una capa intermedia compartida o *middleware* (arquitectura de objetos distribuidos).

Un ejemplo de estas arquitecturas puede ser un sistema de control en tiempo real del tráfico de una ciudad.

Arquitectura entre pares

La **arquitectura entre pares** o ***peer-to-peer*** (P2P) es común en sistemas cooperativos de intercambio de datos, donde múltiples clientes, sin necesidad de un servidor central, pueden compartir información. Estos clientes forman un sistema altamente redundante y tolerante a fallos.

Un ejemplo son las aplicaciones de compartición de ficheros de datos a través de Internet como Emule o BitTorrent.

Arquitectura cliente-servidor (C/S)

La arquitectura cliente-servidor (C/S) es una arquitectura distribuida donde unos nodos "cliente" realizan peticiones y unos nodos "servidor" les dan respuesta. Es decir, el procesamiento de la información se fracciona, a diferencia de la arquitectura monolítica, entre los servidores y los clientes del sistema.

Esta es, probablemente, la arquitectura más conocida y usada de las arquitecturas distribuidas por ser muy intuitiva y, en cierta manera, por emular las relaciones que se establecen entre personas. Es decir, disponen de un cliente –un terminal– que demanda un servicio, y de un servidor, que está especializado en ofrecer dicho servicio, al que responde.

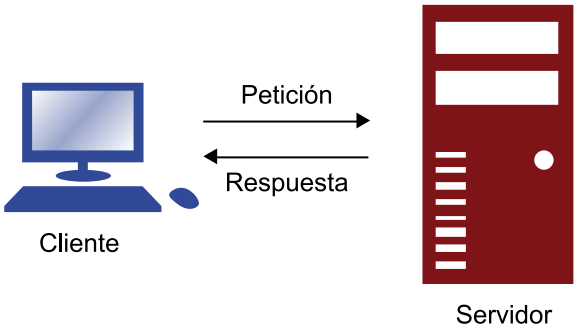

Figura 4. Arquitectura cliente-servidor (C/S)

El **cliente** es el encargado de efectuar dos funciones. Por un lado, interactua directamente con el usuario del sistema a través de, normalmente, una interfaz gráfica. Por otro lado, se comunica con el **servidor** para remitirle solicitudes o peticiones y esperar su respuesta. A su vez, el servidor es el encargado de recibir las peticiones del cliente o de un gran número de clientes, procesarlas y enviar la correspondiente respuesta. Normalmente puede aceptar peticiones de un gran número de clientes. No es frecuente que el servidor interactúe directamente con el usuario del sistema. En una arquitectura C/S podemos tener varios clientes y también varios servidores.

Aunque cliente y servidor podrían residir en una misma máquina (un nivel de arquitectura física), lo habitual en una arquitectura C/S es una estructura de dos niveles de arquitectura física, un nivel cliente y un nivel servidor[3]. Entre estos dos niveles (que pueden estar geográficamente en lugares muy alejados)

[(3)]Por ello en ocasiones a la arquitectura cliente-servidor se llama arquitectura de 2-niveles o 2-*tier*.

se reparten las tres capas de la arquitectura lógica del sistema. En función de la manera como se reparten las tres capas lógicas entre los dos niveles tendremos distintas implementaciones de una arquitectura cliente-servidor.

Una posible implementación es la arquitectura C/S con "clientes pesados" o *fat clients*. En esta implementación, el cliente es el que gestiona la capa de presentación y la capa de negocio (ambas capas están entrelazadas), mientras que la gestión de los datos (capa de datos) queda en el servidor.

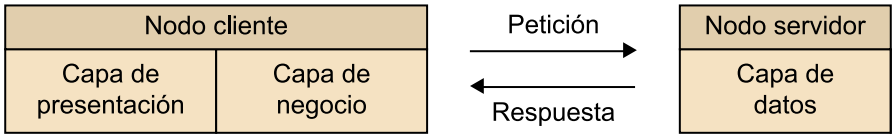

Figura 5. Arquitectura C/S con cliente pesado

Esta implementación permite aprovechar al máximo las capacidades de computación y almacenamiento de los clientes. De esta manera, al trasladar gran parte del trabajo a los clientes, ya no son necesarios los potentes y voluminosos servidores de las arquitecturas monolíticas, a la vez que se continúa garantizando la seguridad e integridad de los datos centralizados en el servidor.

Los sistemas de gestión documental basados en esta implementación presentan un servidor potente que almacena de forma centralizada los documentos en una base de datos y unos clientes pesados, que tienen instalada una aplicación específica de gestión documental, desarrollada en lenguajes como C, Delphi, o Basic, que facilita el acceso a estos documentos.

Esta implementación presenta un claro inconveniente, especialmente si hay muchos clientes en el sistema: cuando es necesario introducir una modificación en la lógica del sistema (capa de negocio) hay que ir cliente a cliente a realizar los cambios (aunque la capa de presentación no cambie). Para superar este inconveniente, y especialmente gracias a la aparición de los navegadores web, podemos plantear una implementación de la arquitectura C/S con "clientes ligeros" o *thin clients*. En esta implementación, el cliente solo gestiona la capa de presentación mientras que la lógica del sistema y sus datos quedan entrelazados en el servidor.

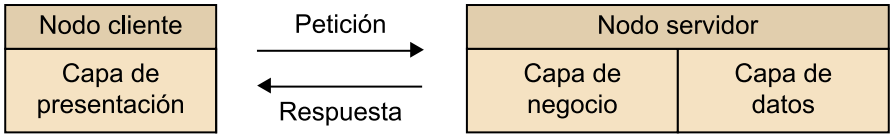

Figura 6. Arquitectura C/S con cliente ligero

Esta implementación permite no tener que instalar un software específico en la parte cliente sino que un simple navegador web (como Internet Explorer, Mozilla FireFox o Google Chrome) es suficiente para acceder a la lógica y a los datos del sistema. Los sistemas basados en esta implementación presentan una parte cliente basada en páginas web (desarrolladas con lenguajes como HTML, JavaScript, Visual Basic Script, Flash o CSS), que se visualizan mediante

cualquier navegador y una parte servidor con toda la lógica del sistema (desarrollada con lenguajes genéricos como Visual C o Visual Basic u otros más específicos como Perl o PHP). Cualquier cambio en la lógica del sistema se realiza solo en el servidor y es automáticamente accesible por todos los clientes. De esta manera, la incorporación de nuevos clientes al sistema es muy simple y hace muy atractiva esta implementación para sistemas con muchos usuarios.

No obstante, el uso de clientes ligeros también presenta inconvenientes. Por un lado, es muy elevada la carga de proceso que soporta el servidor (que realiza todo el trabajo) y también es muy elevado el tráfico de datos que soporta la red de comunicaciones, ya que todos los clientes deben acceder al servidor para ejecutar la lógica del sistema. Por otro lado, se aprovecha poco o nada la capacidad de procesamiento del cliente, ya que el cliente simplemente es una ventana donde visualizar la información gestionada y almacenada en el servidor.

Un intento de maximizar las ventajas y minimizar los inconvenientes de ambas implementaciones es una implementación C/S, donde la capa de negocio se divide entre la parte cliente y la parte servidor.

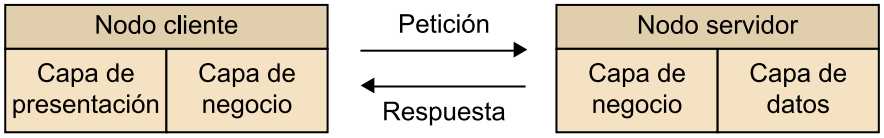

Figura 7. Arquitectura C/S con repartición de capa de negocio

Con esta implementación, si somos capaces de dividir correctamente la capa de negocio (no siempre es fácil ni posible) podemos evitar tener que actualizar todos los clientes ante un cambio en la lógica del sistema. A la vez que aprovechamos la capacidad de computación del cliente (al menos en parte) y reducimos el tráfico de datos a través de la red.

La pregunta que nos podemos plantear es ¿cuál es la mejor implementación de una arquitectura C/S para un sistema de gestión documental, donde necesitamos un servidor central de archivos (documentos y datos asociados) y varios puestos de trabajo (clientes en forma de ordenador de sobremesa, ordenador portátil, tableta o incluso teléfono móvil) que puedan acceder a estos archivos? La implementación elegida condicionará, para bien o para mal, la manera de trabajar de la organización, así que se trata de una decisión importante. La respuesta es que la mejor implementación depende de las necesidades específicas: no es lo mismo una organización de 5 empleados que de 10.000 empleados, ni es lo mismo que los empleados deban acceder al archivo 10 veces al día que una vez a la semana. Y, probablemente, la mejor solución será una implementación que combine las varias posibilidades existentes: para aquellos usuarios (especialmente si son pocos) que deban acceder frecuentemente

al servidor usaríamos un cliente pesado, mientras que para aquellos usuarios (especialmente si son muchos) que deban acceder puntualmente usaríamos un cliente ligero.

Un ejemplo de diferente implementación de una arquitectura C/S en función de las necesidades específicas de un servicio concreto lo encontramos en dos productos de la compañía Google. Por un lado, tenemos el popular servicio de correo electrónico "Gmail" basado en una arquitectura de cliente ligero: el usuario solo necesita un navegador web instalado en su ordenador para visualizar sus correos electrónicos (normalmente archivos pequeños) almacenados en los servidores de Google. Por otro lado, tenemos el servicio de visualización de información geográfica "Google Earth" basado en una arquitectura de cliente pesado: el usuario debe instalarse la aplicación en su ordenador para aprovechar la capacidad de este de almacenar y computar información, ya que la información geográfica implica archivos de datos muy grandes (si utilizáramos clientes ligeros en el caso de Google Earth, la visualización de la información sería extremadamente lenta).

Arquitectura de n-niveles

Las arquitecturas de n-niveles son la evolución de las arquitecturas C/S. En una arquitectura cliente-servidor no existe una verdadera separación entre las diferentes capas lógicas del sistema, ya que siempre hay dos capas que quedan entrelazadas (capas de presentación y negocio en las implementaciones con clientes pesados, y capas de negocio y datos en las implementaciones con clientes ligeros). Las arquitecturas de n-niveles aparecen con el objetivo de establecer una separación real entre cada una de las capas lógicas del sistema: cada capa lógica reside en un nivel físico.

Una característica importante de este tipo de arquitecturas es que cada capa puede acceder solo a las capas inmediatamente adyacentes. Por ejemplo, desde la capa de presentación se puede acceder solo a la capa de negocio pero no a la de datos. De esta manera, se minimiza la dependencia de una capa con las demás. Esta característica permite superar muchas de las limitaciones que presentaba la arquitectura C/S: simplifica en gran medida el desarrollo de cada capa, facilita su actualización, refuerza su seguridad (un usuario no podrá nunca acceder directamente a los datos si no es a través de la capa de negocio), y convierte a esta arquitectura en altamente escalable y flexible.

La implementación más sencilla de este tipo de arquitectura es la de 3-niveles (3-*tier*). En esta implementación encontramos un nivel para la presentación (interfaz de usuario), un nivel para la lógica (también llamado nivel servidor de aplicaciones) y un nivel para el almacenamiento de datos.

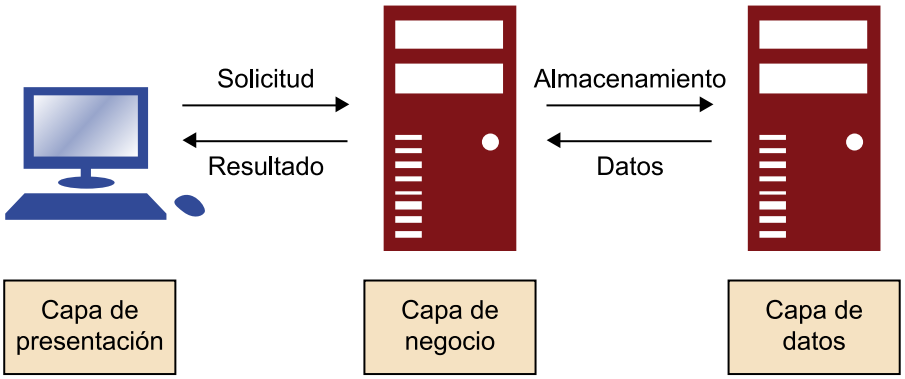

Figura 8. Arquitectura de 3-niveles (3-*tier*)

En el desarrollo de aplicaciones existen dos arquitecturas de 3-niveles muy típicas:

- La **arquitectura J2EE**. Se basa en el lenguaje de programación Java y permite el desarrollo de aplicaciones sobre plataformas (software y hardware) de múltiples fabricantes.

- La **arquitectura .NET**. Permite el desarrollo de aplicaciones exclusivamente para plataformas de Microsoft usando lenguajes de programación como VisualBasic.NET o C#.

Ambas arquitecturas presentan una capa de presentación basada en páginas web, y una capa de negocio y de datos implementadas por clases (modelos o plantillas de código de programación predefinidos) basadas en el lenguaje de programación propio de cada arquitectura. Estas dos arquitecturas ofrecen la posibilidad de desarrollar aplicaciones de manera rápida, robusta y segura, gracias a proporcionar un entorno de trabajo (*framework*) preestablecido, que incluye una metodología de trabajo, librerías de código, programas de soporte, etc.

En un sistema de gestión documental basado en una arquitectura de 3-niveles, diferenciaremos una **capa de datos** donde se almacenan todos los documentos electrónicos del sistema (tanto contenido como metadatos), una **capa de negocio** con todas las funcionalidades de gestión documental necesarias (como crear nuevos documentos, modificar documentos existentes, firmar electrónicamente, definir el cuadro de clasificación, configurar las tablas de evaluación, establecer calendarios de disposición, etc.), y una **capa de presentación** desde la que el usuario podrá acceder a estas funcionalidades para introducir nuevos documentos o consultar los documentos existentes.

A partir de la arquitectura de 3-niveles podemos fragmentar cada una de las capas para obtener una arquitectura de n-niveles (n-*tier*) donde n sea mayor que 3. Las grandes ventajas para un sistema de información que implementa una arquitectura de n-niveles son las siguientes:

- Su escalabilidad, gracias al desacoplamiento entre capas.
- Su mayor control de la seguridad de la información: cada capa puede tener su propia seguridad.
- Su facilidad de actualización: podemos realizar modificaciones en una capa sin tener que modificar las otras.
- La facilidad de su mantenimiento, puesto que se pueden segmentar por capas las actuaciones de mantenimiento tanto correctivas como evolutivas.
- Muy importante, facilita la reutilización de sus componentes, es decir, las capas se pueden estandarizar y ser reutilizadas en otros sistemas.

A modo de ejemplo, una arquitectura típica de n-niveles (en el caso de la figura 9, cinco niveles) es la que divide en dos la capa de presentación y también en dos la capa de datos.

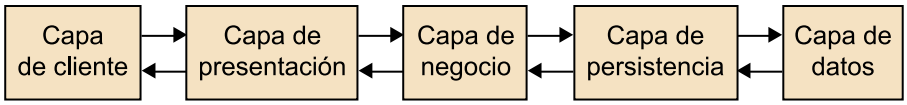

Figura 9. Arquitectura de n-niveles (N-Tier)

Las capas de esta arquitectura de cinco niveles se reparten de la forma siguiente:

- La **capa de cliente**, que interacciona directamente con el usuario y que reside en un cliente muy ligero con un simple navegador web y unas páginas desarrolladas, por ejemplo, en HTML.
- La **capa de presentación**, que contiene la lógica que requiere el usuario, desarrollada, por ejemplo, con lenguajes como J2EE o .NET.
- La **capa de negocio**, con toda la lógica del sistema, desarrollada con alguno de los múltiples lenguajes de programación existentes.
- La **capa de persistencia**, que gestiona todas las lecturas y escrituras de la capa de negocio en la capa de datos a través de tecnologías como ODBC (Open Database Connectivity) o JDBC (Java DataBase Connectivity).
- La **capa de datos**, donde se almacena toda la información en una base de datos como, por ejemplo, Oracle o Microsoft SQL Server.

Pero, como nos podemos imaginar, no todo son ventajas. Una arquitectura de n-niveles se soporta en múltiples elementos hardware que tienen un coste de adquisición, instalación y mantenimiento. Además, estos elementos se comunican entre sí y ello exige unas comunicaciones de datos de alta capacidad

con el coste de adquisición y mantenimiento que conllevan. Si hardware y comunicaciones no se dimensionan correctamente, pueden aparecer cuellos de botella en el sistema y su rendimiento puede verse seriamente afectado.

Ante las arquitecturas de n-niveles, podemos preguntarnos por qué siguen existiendo sistemas, por ejemplo, de gestión documental, implementados sobre una arquitectura C/S. La respuesta es, básicamente, por dos razones: una es el mayor coste que suele representar una arquitectura de n-niveles ante una C/S (este puede ser un condicionante insalvable para organizaciones pequeñas), y otra es el rendimiento y la usabilidad que requiere el usuario del sistema. Si hablamos de rendimiento, no todos los usuarios están dispuestos a tener que esperar, por ejemplo, un minuto para visualizar un documento muy pesado almacenado en el servidor y al que se accede a través de una conexión de Internet de baja capacidad. Si hablamos de usabilidad, los clientes ligeros con un navegador web pueden no dar al usuario una interfaz suficientemente usable a causa de las limitaciones de los lenguajes de programación disponibles para desarrollar interfaces en un navegador web. Una arquitectura C/S con cliente pesado, con una interfaz desarrollada a medida y con capacidad de almacenamiento da respuesta a estos problemas de rendimiento y usabilidad, y por ello sigue siendo actualmente una solución más que válida.

Reflexión

Un excelente ejemplo de interfaz gráfica usable para el usuario son los productos de la compañía Apple. Las opciones son intuitivas y aportan solo aquella información que es imprescindible en el momento en que se necesita. Solo hace falta fijarse en cómo un niño de 7 u 8 años utiliza un iPhone o un iPad sin ninguna ayuda.

Arquitectura SOA

La arquitectura orientada a servicios (*service oriented architecture*, SOA) es una arquitectura de n-niveles con la particularidad de disponer de una capa de servicios donde cada una de las demás capas del sistema "expone" sus funcionalidades (sus servicios) para que puedan ser usadas (o consumidas) por los usuarios del sistema o, incluso, por otros sistemas.

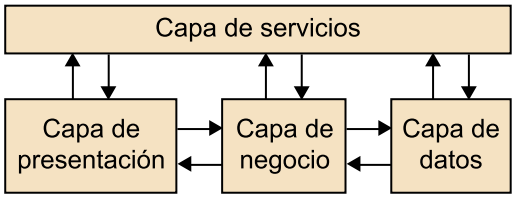

Figura 10. Arquitectura SOA

Un servicio no es más que una funcionalidad del sistema (ya sea de presentación, de negocio o de gestión de datos) encapsulada de manera que sus posibles usuarios (o consumidores) no tengan que preocuparse de las particularidades del hardware y del software en las que se basa (sistema operativo, lenguaje de programación, características de los equipos, etc.). Los servicios en sí se *pueden* desarrollar mediante lenguajes comunes de programación como C, C++ o Java, para posteriormente ser encapsulados con tecnologías basadas en el estándar

XML (*extensible markup language*) y WSDL (*web service description language*), y compartidos mediante el protocolo de comunicaciones estándar SOAP (*simple object access protocol*). La forma más habitual, aunque no la única, de encapsular servicios de manera estándar son los "servicios web" o Web Services (WS). Con la encapsulación de los servicios, toda la complejidad tecnológica que pueda existir en una funcionalidad queda escondida para su usuario.

Una de las principales finalidades de una arquitectura SOA es conseguir la reutilización de servicios preexistentes. Cada sistema de información hace pública la información sobre los servicios expuestos en su capa de servicios para que cualquier usuario autorizado pueda usarlos. Los servicios son publicados en directorios descritos con estándares basados en XML como UDDI (*universal description, discovery, and integration*). En el momento en que nuestro sistema de información requiere de un nuevo servicio, este servicio no lo vamos a desarrollar, sino que en los repositorios de servicios a los que tenemos acceso vamos a buscar un servicio que se adapte a nuestras necesidades. Y como el servicio va a estar basado en tecnologías estándar SOA, lo vamos a poder utilizar sin importarnos las características del hardware y del software que hay detrás de la capa de servicios donde está expuesto.

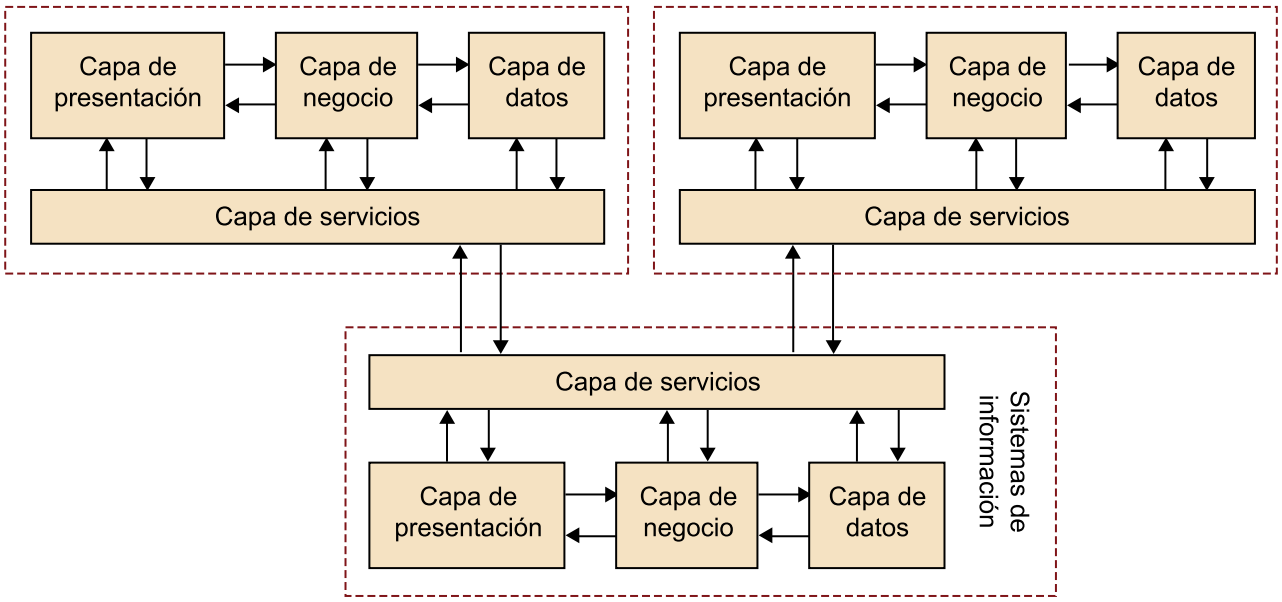

Figura 11. Sistema basado en arquitectura SOA

La principal ventaja de una arquitectura SOA es la capacidad para combinar servicios heterogéneos y con independencia del lenguaje de programación en que han sido escritos. Una determinada funcionalidad de un sistema de información basado en SOA estará formada por varios servicios que serán combinados (orquestados) de una determinada manera para conseguir un resultado concreto[4]. Si un servicio presenta un problema y debe ser sustituido, cambiamos solo este servicio concreto sin tener que tocar el resto de servicios exis-

(4)Para orquestar servicios disponemos de herramientas específicas, como las soluciones BPM que veremos más adelante.

tentes. Todo ello permite sistemas de información muy flexibles. No obstante, ya podemos imaginar que el coste de implantar una arquitectura SOA puede ser elevado.

> Podemos hacer la analogía de una arquitectura SOA con un kit de piezas de construcción para niños. El kit de construcción está compuesto por piezas de diferentes tamaños, formas y colores, incluyendo ruedas, puertas, ventanas, etc. Con estas piezas, un día podemos construir un coche y otro día construir una casa. Ello es mucho más flexible que comprar un coche o una casa de juguete, ya que con el kit de construcción tienes la capacidad de reutilizar las piezas. Además, no hace falta saber de qué están hechas las piezas, simplemente hace falta que encajen unas con otras. En SOA es más o menos lo mismo: las piezas son los servicios, el encaje entre ellas lo garantizan los estándares basados en XML, y las podemos combinar como queramos para construir las funcionalidades que necesitemos en cada momento.

Si pensamos en un sistema de gestión documental, es habitual tener que gestionar documentos que deben ser firmados electrónicamente. Normalmente, la firma electrónica debe ser verificada por un sistema de un tercero (por ejemplo, una agencia de certificación) para que tenga garantías de validez. Si nos encontramos dentro de una arquitectura SOA, el sistema de firma electrónica puede exponer un servicio de validación de firma electrónica que el usuario del sistema de gestión documental puede consumir en el momento en que necesite firmar un documento electrónico. De la misma manera, los servicios de almacenamiento de datos del sistema de gestión documental podrían estar fuera del propio sistema y ser invocados por este solo cuando fuera necesario recuperar o almacenar un documento.

Una implementación posible de una arquitectura SOA es la que utiliza una capa intermedia entre las diferentes capas de servicios de los diferentes sistemas. Esta capa intermedia se llama ESB (*enterprise service bus*). En ella se exponen todos los servicios de las diferentes capas de servicios, de manera que cada sistema solo debe conectarse con la capa ESB para poder consumir todos los servicios disponibles (en vez de tenerse que conectar con todas y cada una de las capas de servicios). Aparte de facilitar la interconexión de servicios, esta capa ofrece funcionalidades de control de errores, securización de la información, trazabilidad, auditorías, y otros servicios de valor añadido para los sistemas de información dentro de la arquitectura SOA.

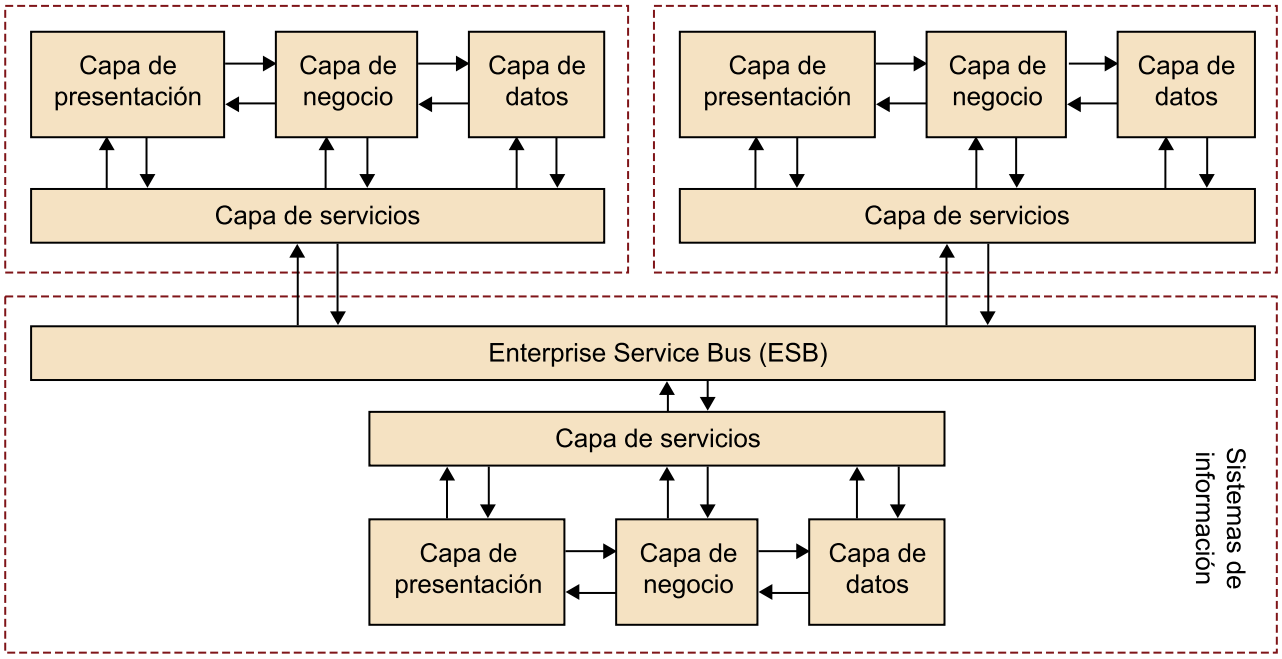

Figura 12. Sistema basado en arquitectura SOA con ESB

El uso de una capa ESB representa un coste adicional de desarrollo y despliegue dentro de una arquitectura SOA, a la vez que puede impactar en el rendimiento del sistema en su conjunto. Ello lo convierte en un elemento que solemos encontrar solo en instalaciones de grandes organizaciones o administraciones públicas.

1.7. *Cloud computing*

No podemos cerrar este punto sin hablar, aunque sea brevemente, de la computación en la nube o *cloud computing*.

La **computación en la nube** es el consumo de recursos informáticos (ya sean hardware o software) ofrecidos como servicios a través de una red de datos (típicamente, Internet).

El *cloud computing* no es una nueva arquitectura de sistemas (aunque algunos lo consideran así) sino que se trata de una arquitectura distribuida donde tenemos la capa de presentación en el lado del usuario mientras que las capas de negocio y de datos del sistema están "en la nube", es decir, en algún lugar del mundo que al usuario le importa poco o nada saber dónde está y al que se accede a través de Internet. De esta manera, a través de un terminal cliente en un ordenador, una tableta o un teléfono inteligente (*smartphone*) conectado a Internet, el usuario accede a los servicios ofrecidos por el sistema, cuya infraestructura hardware y software es responsabilidad y está gestionada por la compañía que ofrece el servicio desde la nube (de forma gratuita o a cambio de una cuota determinada).

La computación en la nube se inicia durante la primera década del siglo XXI cuando grandes compañías de Internet como Google, Amazon o Microsoft construyen enormes centros de procesamiento de datos (CPD o Data Centers, que son grandes edificios que contienen miles de servidores interconectados) para ofrecer sus servicios a través de Internet. Pronto se dieron cuenta de que para ofrecer sus servicios utilizaban normalmente solo una pequeña parte de la capacidad de sus CPD (el resto se usaba solo muy puntualmente para dar servicio a eventos excepcionales) de manera que se les ocurrió que podían alquilar la capacidad sobrante a otras compañías para que pudieran ofrecer sus propios servicios. Posteriormente, muchas compañías han construido sus propios centros de datos para no depender de Google, Amazon o Microsoft. El resultado es la aparición de multitud de servicios basados en *cloud computing*, algunos gratuitos, otros de pago y muchos con una versión básica gratuita y una versión avanzada de pago.

Podemos distinguir tres grandes tipos de servicios de computación en la nube:

- **Infraestructura como servicio** (*infrastructure as a service*, **IaaS**): También llamado *Hardware as a Service* (HaaS), este tipo de servicios ofrecen componentes de hardware (servidores, sistemas de almacenamiento de datos, sistemas de enrutamiento de datos, etc.) que están físicamente en la nube. Este servicio se orienta a arquitectos de sistemas que deseen desarrollar un sistema en su propio entorno de programación y posteriormente instalarlo y ejecutarlo sobre máquinas en la nube. Algunos de los servicios comerciales de IaaS más conocidos son Amazon Elastic Compute Cloud (EC2) o Google Cloud Storage.

- **Plataforma como servicio** (*platform as a service*, **PaaS**): Un nivel por encima de IaaS, este tipo de servicio ofrece plataformas computacionales (hardware y software) en la nube, que incluyen, típicamente, el sistema operativo, un entorno de programación, una base de datos y un servidor web. PaaS está destinado a desarrolladores de aplicaciones que deseen desarrollar y probar sus aplicaciones sin tener que construirse su propia plataforma. Un conocido servicio comercial de PaaS es Google App Engine.

- **Software como servicio** (*software as a service*, **SaaS**): Estos servicios ofrecen aplicaciones completas y listas para ser usadas por un usuario final desde su ordenador, tableta o teléfono. Las aplicaciones ofrecidas con esta modalidad pueden ser de lo más variopintas y muchas de ellas son muy populares. Por ejemplo, aplicaciones de correo electrónico (Google Gmail o Yahoo!Mail), de almacenamiento de datos (Dropbox, Google Drive, o Apple iCloud), de ofimática (Microsoft Office 365 o Evernote), de información geográfica (Google Maps o Google Earth), de mensajería (Twitter), o de ocio (Facebook).

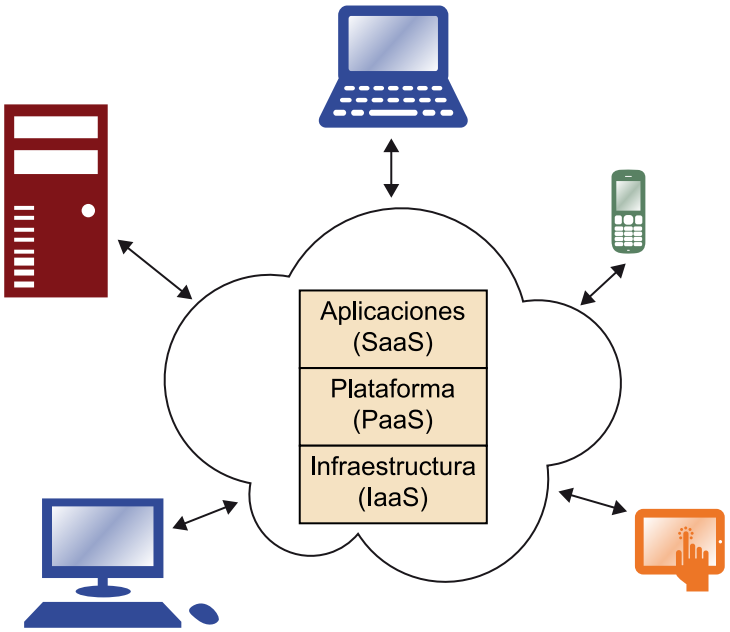

Figura 13. *Cloud computing*

Las principales ventajas del *cloud computing* son evidentes. Como usuarios podemos olvidarnos de tener que construir, mantener y actualizar una infraestructura propia (con los costes, la complejidad y los riesgos que ello conlleva) y disponemos de los servicios de manera ágil y rápida (solo hay que contratarlos), con una gran fiabilidad (los usan cada día miles y miles de usuarios), y con alcance mundial (gracias a Internet). Pero también hay inconvenientes, principalmente dos: la dependencia y la seguridad. Por un lado, al usar servicios en la nube pasamos a depender del proveedor del servicio (que puede decidir cambiar o incluso dejar de prestar el servicio) y, por otro lado, estamos almacenando nuestros datos en unos servidores que no sabemos dónde están y tenemos que confiar que el proveedor del servicio no hará un "mal uso" de ellos (por ejemplo, venderlos a un tercero sin nuestro consentimiento específico). Estos inconvenientes pueden ser banales para un usuario doméstico pero pueden ser una barrera infranqueable para compañías que tratan con datos sensibles, como puede ser una administración pública.

2. Estándares informáticos

En puntos anteriores hemos hablado de varios estándares tecnológicos (como HTML, ODBC, JDBC, XML, SOAP, WSDL, o UDDI), pero ¿qué significa ser un estándar tecnológico y qué implicaciones tiene? En esta unidad vemos qué son los estándares informáticos, por qué y cómo una determinada tecnología se convierte en estándar, qué tipos de estándares podemos encontrar y, por último, detallaremos algunos de los estándares tecnológicos existentes.

2.1. Concepto

Los estándares en el mundo de las tecnologías en general y de la informática en particular son los que permiten que los diferentes sistemas de información puedan entenderse entre ellos e interoperar (intercambiar de forma directa información y resultados). Podemos decir que un estándar es un idioma que los sistemas de información hablan de manera nativa y que les permite que se entiendan entre ellos. No obstante, y al igual que con los idiomas, no existe un único estándar sino que existen multitud de ellos, cada uno aportando unas funcionalidades y unas capacidades distintas.

Un **estándar informático** es un conjunto de acuerdos o normas recogidos en un documento de especificaciones (que debe ser público y accesible de forma gratuita o a un precio simbólico), que actúan como reglas, guías o definiciones de características necesarias para desarrollar software o hardware de sistemas informáticos.

Un ejemplo de estándar de software que todos hemos utilizado (aunque quizá sin darnos cuenta) es el lenguaje de programación web **HTML**, que cualquier navegador de Internet (Explorer, Firefox, Safari, Chrome) sabe interpretar y permite mostrar el contenido de una página web (si no fuera así, una página web debería crearse en el lenguaje propio de cada navegador y eso sería inviable).

Otro ejemplo de estándar de software son los protocolos de comunicación entre ordenadores, como **TCP/IP**, sin los cuales serían imposibles las arquitecturas distribuidas (e Internet no podría existir).

Un ejemplo de estándar de hardware que también todos hemos usado es la conexión USB presente en casi la totalidad de equipos informáticos, independientemente de su fabricante (si no fuera así, un equipo informático debería disponer de varios tipos de conexiones diferentes para poder conectar hardware de fabricantes diferentes).

2.2. Conversión de una tecnología en estándar

Los estándares suelen generarse junto con la aparición de nuevas tecnologías en el mercado. En este momento puede pasar que aparezcan varios candidatos a estándar y que se genere competencia entre ellos hasta que uno prevalezca sobre los demás por razones de flexibilidad y adaptabilidad ante las problemá-

ticas existentes, evolución de la tecnología, facilidad de uso, etc. Pero también puede pasar que, gracias a acuerdos entre proveedores y clientes de una determinada tecnología, se consensúe cuál debe ser el estándar a utilizar.

Un estándar que se impuso a otras opciones gracias a la evolución de la tecnología (mayores velocidades de las redes de comunicaciones) y a la aceptación del gran público es el formato PDF (*portable document format*). Este formato fue inicialmente desarrollado por la firma Adobe Systems en los años 1990, compitió con el formato PostScript (también de la firma Adobe y muy popular en aquellos tiempos) y fue reconocido como estándar ISO 32000-1 en 2008.

Por otro lado, un ejemplo de estándar consensuado entre las diferentes partes implicadas es el estándar de gestión de sistemas de contenidos **CMIS** (*content management interoperability services*). Este es un estándar de **OASIS** (Organization for the Advancement of Structured Information Standards), aprobado en el año 2012, que permite la interoperabilidad de sistemas de gestión de contenidos. La elaboración de este estándar ha contado con la participación y el consenso de varios proveedores de sistemas de gestión de contenidos (que compiten directamente con sus productos en el mercado) como Adobe Systems, Alfresco, EMC, **HP**, Microsoft, OpenText, Oracle o SAP entre otros.

2.3. Tipos de estándares

Los estándares se pueden clasificar, básicamente, en función de su grado de apertura y en función de su carácter legal, y no se deben confundir con los estándares *de facto*.

Desde el **punto de vista de apertura**, encontraremos tres tipos de estándares:

- **Estándar cerrado**: aquel cuya especificación es pública pero existen restricciones (patentes, derechos de autor, marcas, etc.) que impiden que se pueda implementar libremente. Para utilizar el estándar hace falta llegar a un acuerdo particular con su propietario. Identificamos en esta categoría los estándares, por ejemplo, del ECMA (European Computer Manufacturers Association), que es una asociación europea de multinacionales del mundo de la informática.

- **Estándar RAND** (*Reasonable and Non Discriminatory*): aquel cuya especificación es pública y ha sido licenciado bajo unos términos "razonables" y "no discriminatorios" comunes para todo el mercado. Como ya podemos suponer, estos términos pueden ser interpretables y suelen ser fuente de conflicto. Una de las principales críticas a este tipo de estándares es que no permiten la libre distribución del software desarrollado en base a ellos, discriminando así a los modelos basados en programas cuyo código puede ser libremente distribuido (software libre u *open source*). En esta categoría encontramos los estándares de los principales organismos de estandarización como ISO (International Organization for Standardization), IEC (International Electrotechnical Commission) u OASIS (Organization for the Advancement of Structured Information Standards), entre otros.

- **Estándar abierto**: aquel cuya especificación es pública y puede ser implementado sin ninguna restricción. Estos estándares son inclusivos (no excluyen a nadie de su uso) y su utilización es gratuita. La mayoría de soft-

ware libre se basa en este tipo de estándares. Destacamos en esta categoría los estándares de W3C (World Wide Web Consortium), organización sin ánimo de lucro responsable de todos los estándares relacionados con Internet.

Desde el **punto de vista legal**, es importante considerar quién (qué organización) ha emitido el estándar y cuál es el ámbito geográfico de su aplicación (local, estatal, mundial,...):

- Un **estándar legal** es aquel que ha sido adoptado dentro de una directiva o una ley por un gobierno que lo hace obligatorio en entornos públicos, y aquel que ha sido generado por una organización normalizadora, como por ejemplo, CEN (Comité Européen de Normalisation), CENELEC (Comité Européen de Normalisation Electrotechnique) o ETSI (European Telecommunications Standards Institute).

- Un **estándar nacional** es aquel que ha sido normalizado por un cuerpo de estandarización nacional de un país, como puede ser AENOR (Asociación Española de Normalización y Certificación) en España.

- Un **estándar internacional** es aquel que ha sido normalizado por un cuerpo de estandarización internacional formado por representantes legales de cada país, como es el caso de ISO (International Organization for Standardization) o de IEC (International Electrotechnical Commission).

- Un **estándar industrial** es aquel que ha sido normalizado por consorcios industriales que representan una parte importante de una industria, como OASIS, W3C o ECMA.

Es importante destacar que el hecho de que un determinado sistema o formato sea usado de forma muy extendida no lo convierte automáticamente en un estándar tal y como lo hemos definido. En estos casos se habla de **estándar *de facto***, es decir, que son de uso muy común pero que no disponen de unas especificaciones que cumplan con la definición de estándar (un ejemplo es el formato Microsoft Word DOC, ampliamente utilizado sin que ningún cuerpo de estandarización lo haya reconocido). Muchos de los estándares actuales fueron "estándar *de facto*" antes de convertirse realmente en estándar, como el formato PDF o el formato de audio MP3. En el otro extremo, dentro de una organización se puede hablar de formas de trabajar "estándar", pero ello simplemente significa que hay un acuerdo dentro de la organización para realizar las tareas de una determinada manera, sin que realmente exista un estándar como tal. Un estándar requiere, aparte de ser público, que haya sido normalizado y reconocido por algún cuerpo de estandarización.

2.4. Ejemplos de estándares tecnológicos

Algunos de los "estándares" de los que hemos hablado en el punto anterior son:

- Estándares abiertos de **W3C**:
  - **XML** (*extensible markup language*), lenguaje de marcas entendible tanto por un humano como por una máquina.
  - **WSDL** (*web services description language*), lenguaje basado en XML y usado para ofrecer servicios web (método de comunicación entre dos dispositivos a través de Internet).
  - **HTML** (*hypertext markup language*), lenguaje para la creación de páginas web.
  - **SOAP** (*simple object access protocol*), protocolo de intercambio de datos estructurados basado en XML.
- Estándares de **OASIS**:
  - **UDDI** (*universal description, discovery and integration*), registro de recursos disponibles en Internet basado en XML.
  - **CMIS** (*content management interoperability services*), protocolo de interoperabilidad entre sistemas de gestión de contenidos.
- **ODBC** (*open database connectivity*), protocolo para el acceso a una base de datos, es un estándar de SQL Access Group (formado por proveedores de bases de datos).
- **JDBC** (*Java database connectivity*) está basado en ODBC, pero no es un estándar, sino simplemente un protocolo soportado por la compañía Oracle.
- **TCP/IP**, protocolos de comunicación entre ordenadores en los que se basa Internet, es un estándar abierto mantenido por la IETF (Internet Engineering Task Force), que colabora estrechamente con W3C.

3. Bases de datos

Una característica común de todos los sistemas de información es el uso de bases de datos. Por ello, en esta unidad veremos qué son las bases de datos, sus características y criterios que deben tenerse en cuenta para seleccionar un tipo, y los tipos de bases de datos existentes.

3.1. Concepto y características de las bases de datos

Las bases de datos se ubican dentro de la capa de datos de la arquitectura lógica de un sistema y almacenan el que probablemente es su principal activo: los datos del sistema.

Podemos ver una base de datos desde dos puntos de vista, el conceptual y el tecnológico. Conceptualmente, una **base de datos** es un conjunto de información relacionada, que se agrupa o estructura de una determinada manera. Tecnológicamente, una base de datos son una serie de componentes hardware y software encargados de almacenar los datos que va generando un sistema de información.

La parte más interesante de almacenar datos es que de ellos se puede extraer información para poder tomar decisiones de negocio. Pero si los datos no están convenientemente estructurados en la base de datos, esta extracción de información puede no tener las características necesarias para que realmente sea útil a la organización: la extracción puede no ser lo suficientemente rápida, lo suficientemente eficiente o lo suficientemente significativa. Ello nos lleva a la conclusión de que debemos elegir correctamente el tipo de base de datos para cada sistema de información.

Las **características del tipo de base de datos** a utilizar nos vendrán determinadas por los parámetros del sistema de información. Estos son los siguientes:

- La cantidad de datos almacenados.
- El caudal de entrada de datos, es decir qué cantidad de datos nuevos llegan a la base de datos por unidad de tiempo.
- La tipología de los datos, que pueden ser estructurados, no estructurados o una mezcla de ambos.
- El tipo de análisis que se requerirá realizar con ellos, ya que debemos tener en cuenta que no es lo mismo la toma de decisiones en tiempo real, donde

se requieren solo los datos más actuales, que la toma de decisiones estratégicas, donde necesitaremos datos acumulados a lo largo de un periodo de tiempo largo, por ejemplo, para poder identificar tendencias.

Para elegir una base de datos para una organización será muy importante el correcto dimensionamiento de la plataforma hardware que le debe dar soporte y la correcta indexación de la información dentro de la base de datos.

El **rendimiento de la base de datos** en el momento de generar resultados a las consultas del usuario depende de la potencia del hardware asociado (el número de máquinas y su potencia de cálculo) y, especialmente, de cómo de bien o de mal están indexados los datos. La indexación de datos consiste en generar un índice de la información almacenada de manera que sea fácil encontrarla cuando sea requerida (de la misma manera que el índice de un libro nos indica en qué página encontrar el capítulo que nos interesa sin tener que revisar todas las páginas del libro). Una base de datos mal indexada puede suponer minutos de espera para obtener el resultado de una búsqueda frente a los pocos segundos que se requieren si la misma búsqueda se realiza sobre datos correctamente indexados.

Tanto el dimensionamiento de la base de datos como la indexación hacen imprescindibles unas buenas habilidades de los responsables de la administración del sistema de información para dimensionar y estructurar adecuadamente la base de datos. Es habitual contar con expertos del proveedor de la base de datos cuando esta se implanta en una organización para garantizar un rendimiento óptimo.

Un elemento a tener en cuenta en el momento de **elegir una base de datos** es la arquitectura del sistema de información. No todas las bases de datos pueden funcionar sobre cualquier arquitectura. Por ejemplo, en una arquitectura de n-niveles (n-Tier), la base de datos debe poder "desacoplarse" del resto del sistema para poder funcionar en una máquina independiente. Esto es posible en bases de datos como Oracle, Microsoft SQL Server o MySQL, pero no lo es para bases de datos como Microsoft Access o DBase, que no son capaces de funcionar en una máquina separada del resto de máquinas del sistema de información.

3.2. Tipos de bases de datos

Las bases de datos las podemos clasificar en función de cómo estructuran la información en su interior (es decir, su modelo de administración de datos). Esta clasificación nos permite identificar tres grandes tipos de bases de datos: las bases de datos relacionales (que estructuran la información en tablas), las bases de datos orientadas a objetos (cuyo modelo de datos es en base a objetos o "paquetes" de software), y las bases de datos NoSQL (que presentan un modelo sin una estructura definida). Estos tipos los veremos a continuación.

3.2.1. Bases de datos relacionales

Las bases de datos relacionales son bases de datos donde los datos se estructuran en tablas relacionadas entre ellas a través de campos clave.

Por ejemplo, podemos tener una tabla con el identificador de un documento, su descripción (título) y el identificador de la tipología a la que pertenece. En otra tabla, tendríamos todos los diferentes identificadores de tipologías documentales y su descripción. El campo de relación entre ambas tablas sería el código de la tipología documental. Aunque pueda parecer lo contrario, tener la información dividida en dos tablas y establecer relaciones entre ellas a través de campos clave es más eficiente a la hora de explotar los datos que mantenerlos todos en una única tabla.

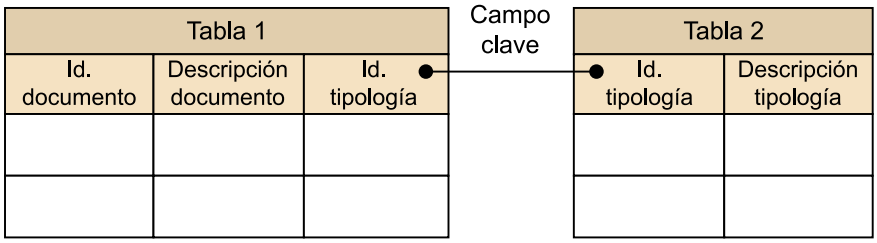

Figura 14

La estructura en tablas relacionadas hace que las bases de datos relacionales sean especialmente recomendables si trabajamos con datos estructurados. Además, en estas bases de datos se puede utilizar el lenguaje estándar (*structured query language*) para la explotación de los datos almacenados (ISO/IEC 9075) SQL.

Los datos en la base de datos relacional se pueden orientar en filas o en columnas. Ello puede parecer poco importante y pasar completamente desapercibido para el usuario final de un sistema de información, pero en función del tipo de datos a almacenar y la manera como se quieran explotar puede tener su importancia. Una base de datos relacional orientada en filas leerá los datos fila a fila, de manera que acceder a los datos de una columna concreta implicará leer todas las filas (con el tiempo que ello implica). Una base de datos relacional orientada en columnas leerá los datos columna a columna, de manera que acceder a los datos de una columna concreta será mucho más rápido (figura 15). Por ello, las bases de datos relacionales orientadas en columnas son más adecuadas para el almacenamiento de grandes volúmenes de datos (muchas columnas) que las orientadas en filas. La base de datos relacional orientada en filas más conocida es Oracle, mientras que la más conocida orientada en columnas es Microsoft SQL Server.

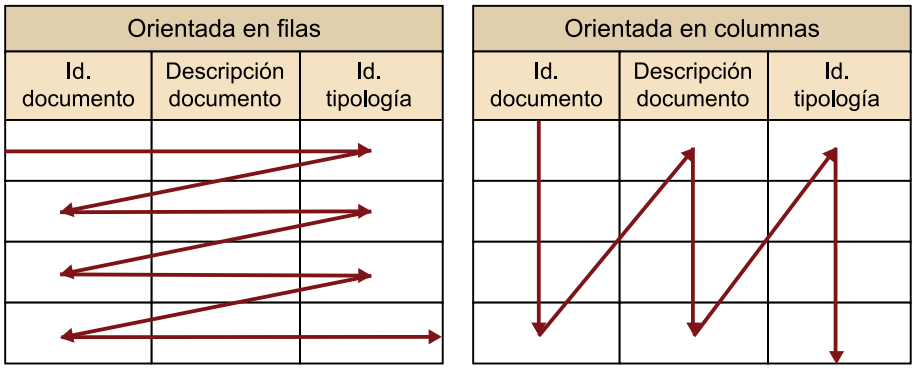

Figura 15. Bases de datos relacionales orientadas en filas y en columnas

La estructura de tablas de una base de datos relacional facilita el mantener la integridad de los datos almacenados, es decir, la garantía de que los datos son consistentes (por ejemplo, evitar tener en una tabla un valor en un campo clave que no exista en su tabla de referencia) y sin errores de formato (por ejemplo, evitar almacenar datos de texto en campos donde se espera un valor numérico). Esta garantía de integridad de los datos es importante cuando necesitamos información fiable para, por ejemplo, la toma de decisiones estratégicas dentro de una organización.

Las bases de datos relacionales son, en general, bastante pesadas para un sistema de información, es decir, requieren de importantes cantidades de esfuerzo (de memoria, de disco, de equipamiento) y tiempo de procesamiento del sistema para mantener las tablas de datos. Por ello es muy importante dimensionar adecuadamente el sistema de información, ya que en caso contrario puede que el tiempo de respuesta ante una petición de datos a la base de datos relacional sea excesivo para el usuario.

3.2.2. Bases de datos orientadas a objetos

De la misma forma que las bases de datos relacionales estructuran los datos en tablas, las bases de datos orientadas a objetos estructuran los datos en paquetes de software llamados “objetos”. Estos paquetes están desarrollados con lenguajes de programación orientados a objetos como Delphi, C++, Visual Basic .NET o Java. Las bases de datos orientadas a objetos están pensadas para el almacenamiento y consulta de datos complejos, como los que puede requerir un software de diseño por ordenador (CAD) o presentaciones multimedia (texto, audio, vídeo, etc.).

La gran ventaja de estas bases de datos es que el lenguaje de programación utilizado para el desarrollo de los objetos puede ser el mismo que se utiliza para el desarrollo del resto del sistema de información (capa de presentación y capa de negocio). De esta manera, la base de datos queda plenamente integrada dentro del sistema de información facilitando su consistencia (en las bases de datos relacionales el modelo de base de datos y el modelo del resto del sistema queda claramente diferenciado).

No obstante, en las bases de datos orientadas a objetos, al contrario que las bases de datos relacionales, no se puede usar el lenguaje estándar SQL. No existe ningún lenguaje que se pueda considerar estándar para la explotación de la información de este tipo de base de datos. Además, determinados productos que utilizan bases de datos orientadas a objetos solo aceptan objetos desarrollados con unos determinados lenguajes de programación y requieren herramientas específicas para su explotación.

3.2.3. Bases de datos NoSQL

El concepto de base de datos NoSQL, como lo entendemos actualmente, aparece en el año 2009. Estas bases de datos están pensadas para dar respuesta a una limitación importante de las bases de datos relacionales (las más utilizadas hasta aquel momento): el almacenamiento y la consulta de grandes cantidades de datos poco o nada estructurados.

Las bases de datos NoSQL están proliferando en los últimos tiempos, en gran medida debido al desarrollo de aplicaciones (sistemas) de Internet que recogen y gestionan cantidades enormes de datos de sus usuarios y que requieren generar resultados en un plazo muy corto de tiempo o incluso de manera instantánea.

> Ejemplos de este tipo de sistemas son los conocidos Facebook, Twitter, Blogger, o Instagram.

Una base de datos relacional es inviable para poder explotar los datos almacenados y generar, por ejemplo, publicidad personalizada para un determinado usuario (en función de los sitios físicos donde se mueve –geolocalización–, las páginas web que visita, las palabras que más frecuentemente utiliza en sus correos electrónicos, etc.) en el momento en que accede a una página web. Una base de datos relacional requeriría de tablas enormes con la mayoría de sus campos vacíos, ofreciendo un rendimiento mediocre en el momento de realizar consultas. Las bases de datos NoSQL no tienen un esquema de datos prefijado y los datos se almacenan en una lista que puede contener información de diferente tipo cada vez.

Desde el punto de vista tecnológico, una base de datos NoSQL es fácil de distribuir entre varios servidores que trabajen en paralelo, de manera que, a pesar de poder manejar cantidades enormes de datos, su rendimiento no tiene por qué verse afectado. Como desventajas de las bases de datos NoSQL, podemos destacar que, al contrario que las bases de datos relacionales, no existe un lenguaje de consulta estándar, y que no garantizan la integridad ni de los datos almacenados ni de los resultados de las consultas que se puedan realizar. La no integridad de los datos puede ser un serio problema si deseáramos generar

información sobre la que tomar decisiones estratégicas para una organización, pero no es excesivamente relevante si lo que necesitamos es generar, por ejemplo, publicidad personalizada para un usuario en una página web.

Por todo ello, las bases de datos NoSQL son perfectas para el almacenamiento y análisis de datos en grandes volúmenes, de una gran variedad y a una gran velocidad (lo que se conoce actualmente como las tres "V" que conforman el concepto de *big data*). La mayor complejidad conceptual y tecnológica que este tipo de bases de datos pueden tener para el usuario se resuelve gracias a los diferentes proveedores que las ofrecen desde "la nube" (*cloud computing*). De esta manera, el usuario no debe preocuparse más que de la explotación de los datos almacenados.

Algunas de las bases de datos NoSQL más populares, la mayoría basadas en software libre (*open source*), son MongoDB, CouchDB, HBase o Riak.

Incluidas dentro de la categoría de bases de datos NoSQL, encontramos las bases de datos XML o las bases de datos orientadas a documento (repositorios documentales) que son especialmente usadas en sistemas de gestión documental. El modelo no estructurado de datos también puede ser aprovechado por soluciones de ERP, CRM, BPM o ECM para beneficiarse de su flexibilidad en el almacenamiento de datos. Hablaremos de qué son y de las principales características de todas estas soluciones en los siguientes puntos.

Reflexión

Ante las ventajas de las bases de datos NoSQL respecto de las bases de datos relacionales, ¿desaparecerán las relacionales? Probablemente no. El modelo relacional es todavía relevante en muchos casos donde la estructura de datos basada en tablas encaja perfectamente. Además, los usuarios están muy familiarizados con las bases de datos relacionales que llevan existiendo muchos años, y existen multitud de herramientas en el mercado para explotar datos mediante el estándar SQL (especialmente muchas herramientas de generación de informes que son las más usadas por los usuarios finales para extraer información de la base de datos). No obstante, todo podría evolucionar en favor de las bases de datos NoSQL, ya que estas ofrecen una manera realmente rápida de desarrollar nuevos sistemas (y esto es muy valorado por los desarrolladores), la mayoría están basadas en software libre (y ello puede significar ventajas económicas), existe una amplia comunidad de usuarios que las hacen evolucionar rápidamente, y, desde el punto de vista tecnológico, están pensadas para trabajar sobre hardware genérico, cosa que les da una gran flexibilidad para ser desplegadas y utilizar toda la potencia del emergente *cloud computing*.

4. Motores de búsqueda

Los sistemas de información, independientemente de su finalidad, pueden almacenar datos en grandes cantidades dentro de sus bases de datos. Pero el almacenamiento de datos es solo la mitad del trabajo necesario para convertir estos datos en uno de los principales activos del sistema. La otra mitad es la capacidad de recuperar los datos almacenados de manera rápida en el momento en que sean requeridos. La recuperación de los datos del sistema puede ser una tarea menor cuando se trata de poca cantidad de información, pero cuando la cantidad de datos es muy grande su recuperación se convierte en un problema de vital importancia para la viabilidad del sistema. Es decir, de nada sirve tener mucha información si no somos capaces de recuperar aquello que buscamos en el momento en que lo buscamos.

Por ello, un tema importante que también debemos tener en cuenta son los instrumentos para buscar y recuperar de manera eficiente la información contenida en las bases de datos.

En esta unidad veremos qué son los motores de búsqueda y sus tipos.

4.1. Concepto de los motores de búsqueda

Los **motores de búsqueda** o buscadores son herramientas específicamente diseñadas para buscar datos almacenados dentro de las bases de datos de un sistema de información.

Aunque el concepto de motor de búsqueda se relaciona normalmente con los buscadores de Internet, un motor de búsqueda es una herramienta genérica aplicable en cualquier sistema de información. Es habitual que los sistemas de información incorporen un motor de búsqueda. Los buscadores suelen partir de una palabra o palabras clave para generar una lista de resultados que contienen o están de alguna manera relacionados con estas palabras clave.

Los motores de búsqueda más conocidos y que todos hemos utilizado son los buscadores de Internet con el buscador de Google (www.google.com) a la cabeza, aunque existen decenas de ellos como Bing (www.bing.com), Yahoo! (www.search.yahoo.com), DuckDuckGo (duckduckgo.com/), Baidu (www.baidu.com), Open Directory Project (www.dmoz.com), etc.

Estos buscadores facilitan encontrar una información concreta entre la cantidad gigantesca de datos heterogéneos y altamente distribuidos que contiene Internet de una forma extremadamente rápida (pensemos por un momento la ardua tarea que representaría encontrar información en Internet sin los buscadores...).

Un motor de búsqueda es también imprescindible en un sistema de gestión de documentos electrónicos donde recuperar documentos (tanto a partir de su contenido como a partir de sus metadatos) es una de las funcionalidades clave del sistema. El buscador será especialmente útil cuando el sistema de gestión documental se alimente de varios repositorios de documentación federados (los contenidos de los documentos están almacenados en repositorios independientes entre sí, pero deben poder ser visualizados como si estuvieran en un único repositorio). Sin el buscador, ser capaces de encontrar un documento electrónico dentro de un repositorio con centenares de miles de expedientes electrónicos de, por ejemplo, una administración pública sería una tarea larga y tediosa.

4.2. Métodos de organización e indexación empleados por los motores

La clave de un motor de búsqueda es cómo consigue obtener la información, organizarla e indexarla de manera que sea fácil de recuperar cuando sea requerida. Existen básicamente dos métodos para resolver este punto:

- **Arañas** (***spiders***, también llamados ***crawlers*** o **robots**): Son programas que inspeccionan de forma automática y sistemática la información que contiene el sistema. Esta inspección se realiza con una determinada frecuencia para detectar posibles cambios y actualizaciones en la información del sistema. El programa recopila los datos que considera oportunos. Por ejemplo, puede inspeccionar el contenido de páginas web si se trata de Internet o el contenido o los metadatos de documentos electrónicos si se trata de un sistema de gestión documental, etc. Los datos recopilados pueden ser más o menos voluminosos en función de las necesidades del usuario, de la capacidad del buscador y de los algoritmos de búsqueda que se vayan a aplicar. Una vez recopilados los datos, la araña los almacena e indexa (crea un índice de todos los elementos explorados) en una base de datos propia, y ofrece al usuario la explotación de esta base de datos (la búsqueda). Cuando se realiza una consulta, el motor de búsqueda localiza en el índice los elementos deseados generando los resultados rápidamente.

  El ejemplo más conocido de este método es el buscador de Google (www.google.com).

- **Directorios**: Se trata de recoger la información del sistema (de manera manual o con un mínimo de automatización) que es revisada por personas y posteriormente clasificada en categorías. Al contrario que con las arañas, en este caso no se almacenan contenidos sino simplemente etiquetas que

se colocan sobre la información del sistema siguiendo algún tipo de clasificación (por ejemplo, por temática) y que se usarán para que el usuario del buscador pueda localizar la información cuando lo requiera.

Un ejemplo de este método es el Open Directory Project (www.dmoz.org).

Estos dos métodos implican dos filosofías opuestas de afrontar la recolección y búsqueda de los datos de un sistema de información. Con las arañas (una tecnología cara y compleja) no importa que el sistema evolucione sin ningún tipo de patrón concreto, ya que los programas de rastreo harán su trabajo independientemente de cómo esté organizado el sistema. Por el contrario, el método de los directorios (una tecnología barata y simple) pretende que el sistema se organice de una forma concreta (se autoorganice) para evitar que los resultados que se puedan obtener con una búsqueda no tengan sentido. En cualquier caso, actualmente el método de mayor aceptación, a pesar de sus costes, es el de las arañas.

5. Sistemas de gestión de datos

En puntos anteriores, hemos definido un sistema de información como el conjunto de elementos que actúan de manera conjunta para capturar, procesar, almacenar y distribuir datos e información con una determinada finalidad. Esta definición sitúa como eje principal de un sistema, independientemente de la arquitectura sobre la que esté implementado, los datos (información sin procesar) y la información (datos procesados). La capa de presentación del sistema nos permitirá capturar datos introducidos por el usuario, la capa de negocio los procesará en base a las reglas del sistema y la capa de datos los almacenará. De la misma manera, la capa de negocio podrá recuperar datos de la capa de datos y mostrarlos al usuario a través de la capa de presentación.

En cualquier caso, es de vital importancia que el sistema pueda garantizarnos que los datos que gestiona tengan cinco características:

- **Calidad**: Los datos deben ser íntegros y verdaderos para poder ofrecernos un fiel reflejo de la realidad, es decir, deben ser datos con un buen nivel de calidad.

- **Cantidad**: Los datos nos deberán ayudar a tomar decisiones de diferente índole dentro de nuestra organización, por lo que debemos disponer de una determinada cantidad de ellos para poder tomar decisiones precisas y acertadas.

- **Relevancia**: El sistema nos deberá ofrecer solo aquellos datos que sean relevantes en cada momento, ya que aquellos que no lo sean no nos aportarán ningún valor.

- **Oportunidad**: Debemos tener los datos disponibles cuando los necesitemos, ni antes ni después, para evitar perder la oportunidad de tomar la decisión correcta.

- **Estructura**: Los datos deben tener una determinada estructura para que tanto el sistema como el usuario puedan comprenderlos fácilmente.

A lo largo del tiempo se han desarrollado multitud de sistemas para la gestión de datos dentro de una organización. Estos sistemas se han ido especializando hasta conformar soluciones (que incluyen tanto hardware como software), que podemos identificar bajo siglas como ERP, CRM, BPM, BI, o ECM, que estudiaremos en esta unidad. Debemos tener en cuenta, sin embargo, que es-

tas siglas no engloban solo elementos tecnológicos, sino que representan una determinada estrategia de gestión de la información (metodologías, políticas, etc.) para abordar la mejora de una organización.

Reflexión

Es importante resaltar que los elementos tecnológicos que se incluyen en un sistema ERP, CRM, BPM, BI o ECM son elementos que están al servicio del negocio para ayudarlo en todo lo que sea posible. Pero estos elementos por sí solos nunca podrán resolver los problemas de negocio de una organización. En la resolución de un problema de negocio puede ser necesaria la tecnología, pero es imprescindible que haya una estrategia, una metodología, unos recursos, etc.

Las siglas ERP, CRM, BPM, BI, o ECM se han convertido en siglas de uso común y es muy probable que nos las encontremos cuando abordemos la necesidad de un nuevo sistema de información. Dentro de cada una de ellas, podemos encontrar muchas soluciones de mercado, tanto basadas en software propietario como en software libre, para instalar dentro de nuestra propia organización como para explotar desde "la nube". Citaremos algunas de las soluciones de mercado más representativas de cada tipo de sistema en los siguientes apartados.

Información complementaria

Para conocer cuáles son las soluciones de ERP, CRM, BPM, BI, o ECM mejor valoradas en el mercado, podemos recurrir a estudios realizados por compañías consultoras y de investigación de las tecnologías de la información como Gartner (www.gartner.com/) o Forrester (www.forrester.com). Gartner elabora periódicamente y para cada tipo de tecnología un "cuadrante mágico", que es una representación gráfica donde identifica los "líderes" (soluciones con mayor potencial y alcance), los "aspirantes" (soluciones con potencial pero sin el alcance de los líderes), los "visionarios" (soluciones con potencial limitado pero alcance amplio), y los "nichos específicos" (soluciones enfocadas a problemáticas específicas). De la misma forma, Forrester elabora su "Forrester Wave" y también clasifica entre cuatro categorías (*leader*, *strong performer*, *contender* y *risky bet*) cada una de las soluciones.

Los sistemas ERP, CRM, BPM, BI y ECM se basan en arquitecturas distribuidas y persiguen, cada uno a su manera, objetivos comunes dentro de una organización, como aumentar su eficiencia, mejorar su productividad, reducir sus costes, aprovechar economías de escala, homogeneizar la manera de trabajar, aumentar la coordinación, automatizar procesos de negocio, o hacer más transparentes sus operaciones. Todos estos sistemas tienen en su corazón una base de datos en la que se almacena la información que se va a explotar para aportar valor a la organización. Actualmente, estas bases de datos suelen ser relacionales u orientadas a objetos, pero en la mayoría de casos son combinación de ambas junto a bases de datos propias de cada sistema.

5.1. ERP (*enterprise resource planning*)

Un **ERP** es un sistema de información que tiene por objetivo aglutinar y gestionar todos los datos de las diferentes áreas de una organización.

Con este tipo de sistema, el usuario puede obtener una visión integral de la organización a través de una única aplicación de usuario que centraliza toda la información. Ello permite estandarizar los procesos de negocio internos de la organización (típicamente procesos de gestión financiera, gestión de recursos humanos, gestión de la producción de productos o servicios, gestión de compras y ventas, y gestión de clientes) garantizando que la información se mantiene estructurada, íntegra y lista para ser de utilidad a las diferentes áreas de la misma. Esta estandarización es especialmente importante en organizaciones grandes con muchos empleados, y es por ello por lo que este tipo de organizaciones son las que más utilizan sistemas ERP.

La arquitectura de implementación de este tipo de sistemas suele ser una arquitectura cliente-servidor. Típicamente, la implementación de la capa de datos es una única base de datos relacional que actúa como almacén centralizado de toda la información del sistema. Esta centralización de los datos garantiza su integridad entre las diferentes áreas de la organización, pero también añade complejidad a su explotación por parte de los usuarios y de los procesos de negocio (que pueden involucrar diversas áreas en un mismo proceso).

La principal característica de estos sistemas es su modularidad. Un ERP está formado por una serie de módulos independientes entre sí pero que comparten información entre ellos a través de la base de datos central. Algunos de los módulos principales de un ERP son: finanzas y contabilidad, recursos humanos, producción, logística, compras y ventas, y clientes. Esta modularidad del sistema permite partir de unos módulos básicos y tener la posibilidad de ir añadiendo módulos más específicos a lo largo del tiempo (como, por ejemplo, un módulo específico de gestión documental, un módulo de gestión de la calidad, un módulo de gestión del mantenimiento, etc.).

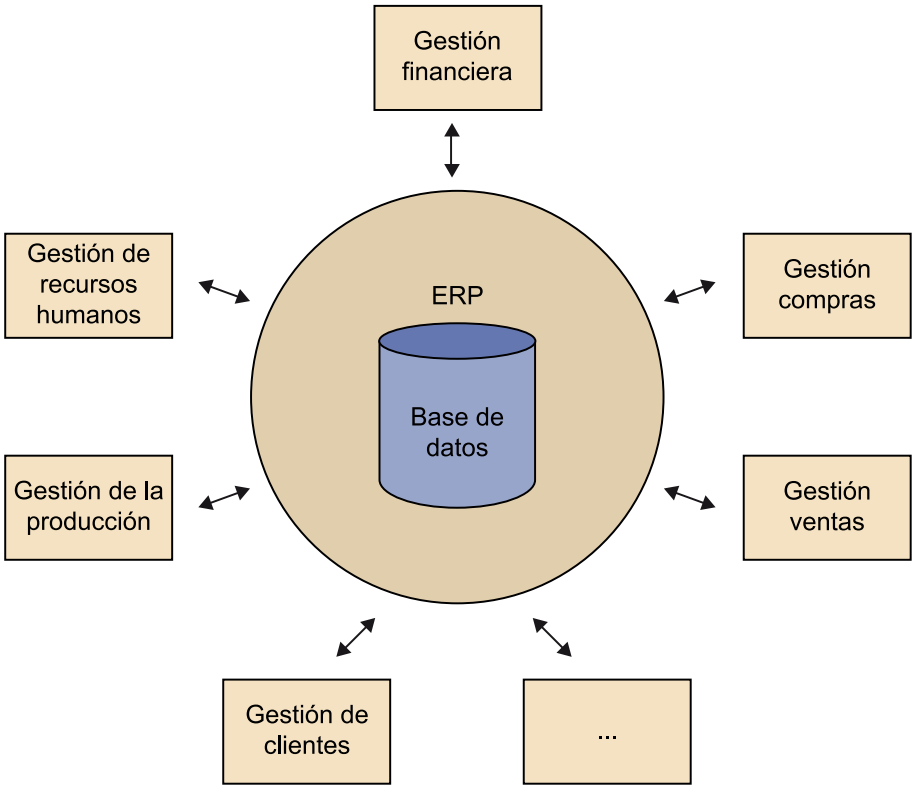

Figura 16. ERP

La base de datos central de un ERP permite al usuario obtener una visión unificada del negocio con información que involucra a todas las áreas de la organización. Los empleados de toda la organización pueden buscar, almacenar y compartir información a través de la base de datos central de manera eficiente (minimizando las transferencias de datos y evitando la información duplicada o la necesidad de entrar varias veces la misma información). A su vez, los miembros de la dirección disponen de una visión única de la organización que les permite encontrar oportunidades para mejorar la eficiencia y la productividad. Y este es el valor real y distintivo que aporta un ERP a una organización: la posibilidad de guiar la estrategia corporativa.

La implementación de un ERP requiere un esfuerzo económico y humano importante. Estos sistemas suelen ser paquetes de software "cerrados" (hacen lo que el fabricante ha decidido que hagan y establecen procesos de negocio predefinidos) aunque en la mayoría de los casos los podemos (y los debemos) adaptar a nuestra organización a través de herramientas de desarrollo que se adquieren, normalmente, con el propio ERP. Esta necesidad de adaptación implica que la organización deba contratar a un implementador que lo ponga en funcionamiento (con su coste económico y temporal asociado). En cualquier caso, debido a su transversalidad en la organización, el proceso de selección de un ERP debe contemplar los requerimientos de todas las áreas de la organización, asegurar la escalabilidad del sistema, y, sobre todo, conseguir un sistema que se adapte a la organización, evitando que la organización deba adaptarse al sistema. Por todo ello, será imprescindible una correcta formación y capa-

citación de los usuarios del sistema para obtener, usar y organizar la información necesaria, así como la correcta integración del ERP con los sistemas ya existentes en la organización.

Actualmente existen multitud de sistemas ERP en el mercado. Algunos de los más conocidos son:

- SAP (www.sap.com/spain/solutions/business-suite/erp/index.epx)
- Microsoft Dynamics ERP (www.microsoft.com/es-es/dynamics/erp.aspx)
- Oracle JD Edwards EnterpriseOne (www.oracle.com/us/solutions/midsize/enterprise-resource-planning/overview/index.html)

5.2. CRM (*customer relationship management*)

Un **CRM** es un sistema de información cuyo objetivo es gestionar la información de los clientes de una organización.

De la misma manera que un ERP gestiona la información dentro de la organización centrándose en los recursos y la productividad de la misma (sistema *organization-oriented*), un CRM gestiona la información sobre los actuales y también sobre los potenciales clientes de la organización (sistema *customer-oriented*). Como ya podemos intuir, los principales usuarios de este tipo de sistema son el departamento de marketing, el departamento comercial (o de ventas) y el departamento de atención al cliente de una organización.

Un CRM no es solo una solución de software sino que engloba la metodología, la disciplina y la tecnología para automatizar y mejorar los procesos de relación entre la organización y sus clientes. El objetivo de un CRM es aglutinar el máximo número de datos acerca de los clientes de la organización. Sobre estos datos podremos aplicar técnicas y herramientas para identificar patrones de comportamiento y así poder conocer al detalle las necesidades de nuestros clientes. De esta manera podremos, por ejemplo, personalizar los servicios que les ofrecemos, es decir, no vamos a ofrecer lo mismo a todos los clientes sino que a cada cliente le ofreceremos solo las ofertas y los servicios enfocados a sus necesidades específicas. Todo ello debe permitir a la organización una reducción de gastos en publicidad y marketing y, especialmente, una mejora en la calidad del servicio ofrecido al cliente (que se traduce en un cliente más satisfecho).

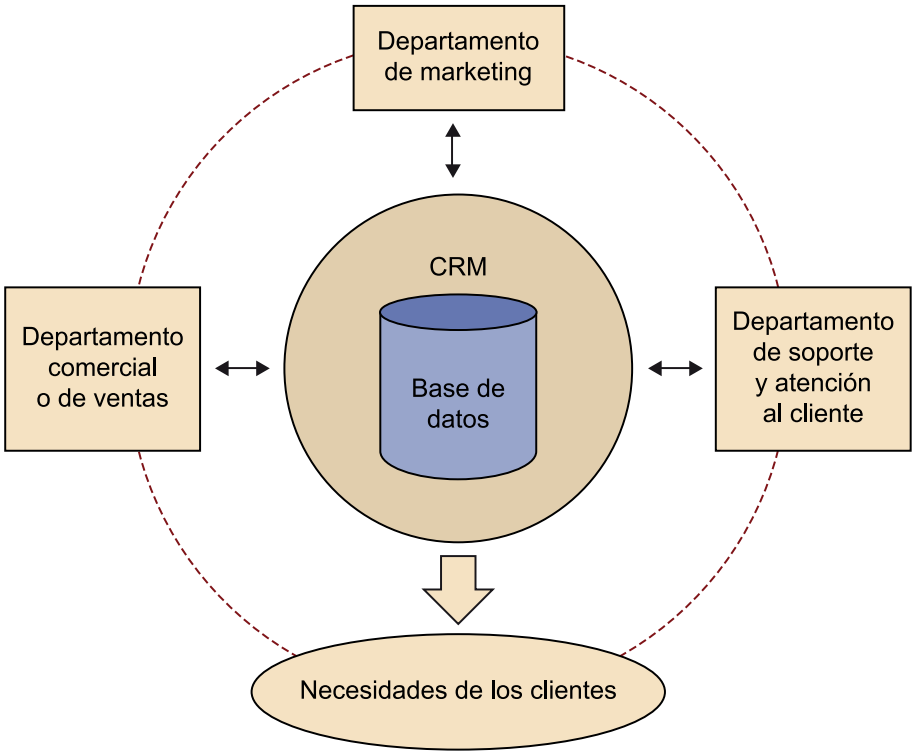

Figura 17. CRM

Reflexión

En la última década, la globalización ha generado un mercado global tremendamente competitivo que facilita a los consumidores cambiar de una compañía a otra para proveerse de los distintos productos y servicios que necesitan. Ya no es suficiente intuir lo que necesita un cliente, es necesario saber qué quiere. Este es el principal motivo de la aparición y proliferación de los sistemas de CRM. Si usamos de forma efectiva un CRM, podemos construir una relación con nuestro cliente que dure toda la vida, y esto, en la era de la globalización, no tiene precio.

Las características de un sistema CRM son muy similares a las de un ERP: típicamente una arquitectura cliente-servidor con una base de datos centralizada. Funcionalmente, y también de manera similar a un ERP, un CRM ofrece procesos de negocio predefinidos que obligan a su adaptación en el momento de implantar el sistema en una organización. Aunque un CRM puede presentarse como un sistema de información independiente, es habitual que se incluya como uno de los módulos, concretamente el módulo de clientes, de un ERP.

La implantación de algún tipo de solución de CRM es condición necesaria pero no suficiente para garantizar una correcta relación con los clientes. Es imprescindible que la solución se combine con los objetivos y las estrategias de negocio de la organización para garantizar que estén alineados y trabajando en la misma dirección. Además, la implantación de un CRM probablemente dará excelentes resultados a largo plazo pero no se pueden esperar grandes avances en el corto plazo, ya que primero hará falta recopilar y analizar gran cantidad de información acerca de los clientes.

Algunas de las soluciones de CRM más conocidas son:

- salesforce.com (www.salesforce.com/), que se caracteriza por ser ofrecido en modalidad SaaS en la nube.
- Oracle Siebel CRM (www.oracle.com/us/products/applications/siebel/overview/index.html)
- Microsoft Dynamics CRM (crm.dynamics.com/es-es/home)
- SAP (www.sap.com/spain/solutions/business-suite/crm/index.epx)

5.3. BPM (*business process management*)

Todas las organizaciones utilizan procesos para conseguir alcanzar sus metas de negocio y muchas de ellas consideran estos procesos como activos fundamentales para su estrategia de negocio (al igual que se puede considerar como activo a un edificio o a una máquina).

Un proceso de negocio es un conjunto de tareas interrelacionadas, que se orientan a lograr un objetivo de negocio generando un resultado de valor para el cliente del proceso.

Para poner un ejemplo, pensemos en una organización que establece como objetivo de negocio la eliminación del papel de todas sus actividades. Para poder alcanzar este objetivo, la organización deberá establecer procesos de gestión documental para asegurar que los documentos de la organización nacen, viven y mueren electrónicos, conservando todas las características de autenticidad, integridad, fiabilidad y disponibilidad. Algunos de los procesos que se podrían requerir son:

- Procesos de creación de documentos para obtener documentos en formato electrónico desde el momento de su nacimiento: qué plantillas utilizar, qué formatos definir, qué metadatos incluir,...
- Procesos de digitalización de documentos para transformar a formato electrónico aquellos documentos de la organización que se han generado en formato papel: qué mecanismos utilizar, qué características de digitalización aplicar, cómo garantizar la autenticidad del resultado del proceso,...
- Procesos de almacenamiento y recuperación de documentos electrónicos para garantizar que los documentos electrónicos pueden ser consultados y modificados por las personas autorizadas.
- Procesos de conservación de documentos electrónicos para garantizar su integridad y autenticidad a lo largo del tiempo.
- Procesos de eliminación de documentos electrónicos cumpliendo con los requisitos legales y normativos que puedan aplicar.

Un proceso de negocio puede durar desde unos pocos minutos hasta varios días o incluso meses, dependiendo de su complejidad y de la duración de las tareas que lo constituyen. Por ejemplo, la inclusión de una firma electrónica en un documento es un proceso que dura segundos, mientras que la construcción de un avión requiere de un proceso de meses.

Tradicionalmente, el modelo de negocio de una organización se centraba en su estructura organizativa, es decir, cada área de la organización (finanzas, recursos humanos, marketing, operaciones,...) desarrollaba y ejecutaba sus propios procesos de forma independiente sin considerar los procesos de otras áreas. El modelo de negocio que ha surgido en las últimas décadas es un modelo centrado en los procesos, es decir, los procesos son de la organización y transversales a todas sus áreas y deben ser gestionados en consecuencia.

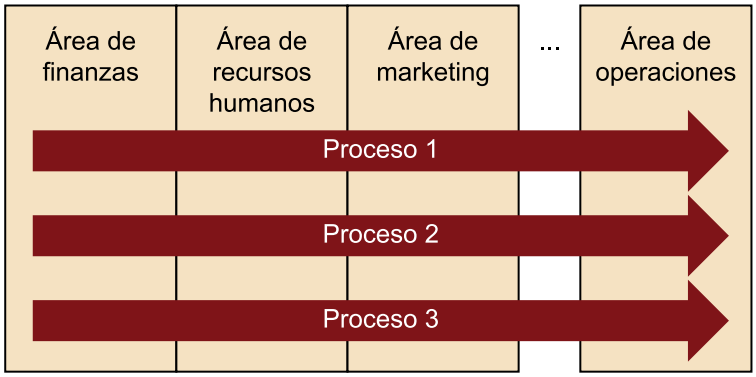

Figura 18. Procesos transversales a la organización

Es en este planteamiento donde toman importancia las herramientas de gestión de procesos. Los sistemas BPM ponen a nuestra disposición la posibilidad de diseñar, administrar y mejorar a lo largo del tiempo los procesos de negocio de una organización. Al igual que el ERP se orienta a la organización y el CRM se orienta al cliente, BPM se orienta a procesos (*process-oriented*).

BPM es una disciplina que utiliza métodos, políticas, tecnologías y prácticas de gestión para administrar los procesos de negocio de una organización.

Con esta definición ya podemos entender que BPM no es solo tecnología sino que hay una parte muy importante de gestión. No obstante, todo BPM se apoya sobre unas herramientas tecnológicas específicas donde almacenar y gestionar los datos referentes a los procesos de negocio.

El objetivo de los sistemas BPM es sencillo de plantear, aunque puede ser muy complejo de llevar a la práctica. Un BPM extrae los procesos de la organización y los modela, incluyendo todas sus relaciones y condicionantes, de manera que el modelo de procesos resultante sea completamente independiente de las tecnologías necesarias para su implementación. De este modo, el modelo de procesos se convierte en un activo más de la organización. Las soluciones de BPM nos permitirán visualizar y administrar de manera fácil y rápida este modelo y, sobre todo, nos permitirán enlazar el modelo con todos los elementos (tecnológicos y no tecnológicos) que deben permitir ejecutarlo y monitorizarlo en el día a día de la organización.

Aunque en general se habla de soluciones o herramientas de BPM, podemos diferenciar claramente tres tipos de herramientas (que pueden formar parte de un único paquete de software o pueden ser soluciones independientes):

- Herramientas de **BPA** (*business process analysis*) para el análisis, modelización, diseño y simulación de procesos. Estas herramientas permiten generar el modelo de procesos de una organización. Para ello, se ha desarrollado el lenguaje BPMN (*business process modeling notation*) como estándar (*de facto*) para la diagramación de procesos de negocio que permite dibujar unos modelos comprensibles tanto para los analistas de negocio (quienes crean los modelos de los procesos) como para los desarrolladores (responsables de implementar los sistemas y tecnologías para los procesos), y los responsables de negocio (quienes monitorizarán y gestionarán los procesos).

- Herramientas de **BPM** (*business process management*) para la automatización y orquestación de los procesos de una organización. Estas herramientas son los motores de las reglas de negocio, es decir, son las que a partir del modelo de un proceso permiten su implementación y su ejecución. El lenguaje estándar (de OASIS) para estas herramientas es BPEL (*business process execution language*) y la gran ventaja es que puede ser generado a partir de un modelo de procesos realizado con BPMN.

- Herramientas de **BAM** (*business activity monitoring*) para la monitorización en tiempo real de la información de los procesos de una organización. Estas herramientas deben permitir entender cómo de bien o de mal se alinean los procesos con los objetivos de negocio y permitir la mejora continua de estos procesos.

Las herramientas de BPM configuran el motor de reglas del sistema de información (capa de negocio de la arquitectura lógica) y permiten la gestión de procesos de negocio que pueden involucrar múltiples áreas y múltiples colectivos de usuarios de la organización (cada uno con sus particularidades específicas). A partir de un determinado evento (por ejemplo, la necesidad de firmar un documento electrónico), BPM desencadena la ejecución de un proceso (por ejemplo, la comprobación e inclusión de la firma electrónica en el documento). La ejecución del proceso va llamando (orquestando) paso a paso a todos los elementos necesarios para la correcta gestión del evento. En cada paso de la ejecución harán falta datos "de entrada" (que pueden proceder del usuario del sistema que los introduce a través de la capa de presentación o que pueden ser recuperados de la base de datos de la capa de datos del sistema) para generar los datos "de salida" (que, a su vez, pueden ser datos "de entrada" del siguiente paso). La ejecución irá generando alarmas y notificaciones cuando sea necesario para que el usuario del sistema pueda tomar las decisiones oportunas.

Las soluciones de BPM han alcanzado gran relevancia especialmente a partir del auge de la arquitectura SOA. Ello es debido a que estas soluciones no solo pueden orquestar elementos del propio sistema de información al que pertenecen, sino que pueden también llamar y ejecutar servicios ofrecidos por otros sistemas de información.

Imaginemos, por ejemplo, el proceso para firmar electrónicamente un documento electrónico dentro de un sistema de gestión documental. Ante la necesidad de firmar un documento electrónico (evento), la solución de BPM lanza y monitoriza un proceso que:

1) Recupera el documento electrónico a firmar de la base de datos donde está almacenado.

2) Recupera los datos del certificado digital que el usuario que desea firmar el documento ha colocado en el lector de certificados de su ordenador (si no lo ha colocado se genera una notificación indicando al usuario que lo coloque).

3) Llama y envía los datos del certificado del usuario al servicio de comprobación ofrecido por el sistema de información de la entidad emisora del certificado (servicio externo a nuestra organización).

4) Si la respuesta del servicio de comprobación es afirmativa, se incluye la firma electrónica en el documento. En caso contrario, se informa al usuario del problema.

De esta manera, se combina la automatización de los procesos de la organización con la posibilidad de explotar servicios de diferentes proveedores que ofrece la arquitectura SOA. La combinación de soluciones BPM con arquitecturas SOA consigue un alto rendimiento de los procesos de negocio de forma segura, coherente y preservando la integridad de la información. Además, permite una mejor utilización de los recursos de la organización, una mayor capacidad de reacción ante los cambios del negocio (agilidad), y un ahorro de costes de tecnologías de la información.

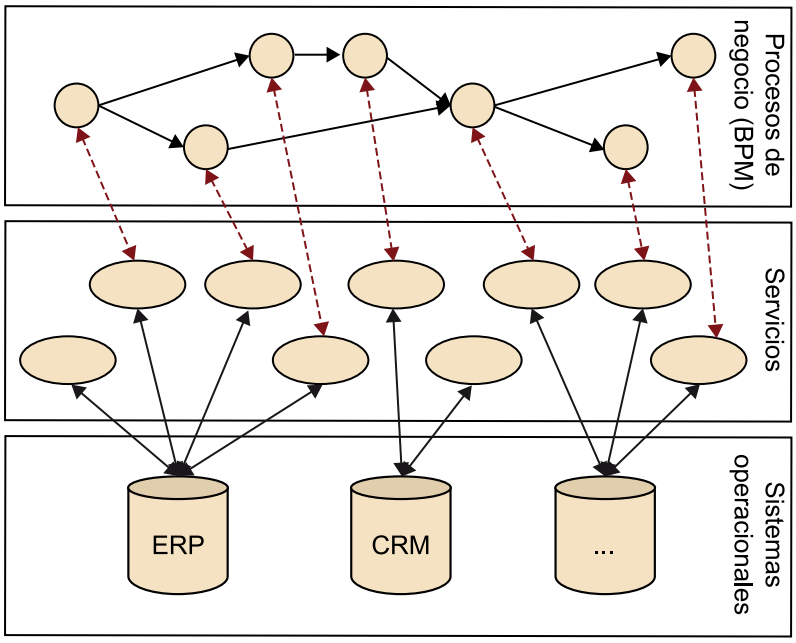

Figura 19. BPM y SOA

En definitiva, BPM permite alinear los objetivos de negocio con la infraestructura tecnológica de la organización y hacer que esta alineación sea flexible y se adapte de manera continua a los cambios del negocio.

Algunas de las suites de productos de BPM más completas del mercado son las siguientes:

- IBM (http://www-03.ibm.com/software/products/us/en/category/BPM-SOFTWARE)
- Microsoft (http://www.microsoft.com/es-es/biztalk/default.aspx)
- Oracle (http://www.oracle.com/us/technologies/bpm/overview/index.html)

5.4. BI (*business intelligence*)

El concepto de *business intelligence* se refiere a las técnicas y los procesos de acumulación, evaluación y presentación de datos (de la propia organización, de la competencia, del mercado, etc.) con la finalidad de poder mejorar la toma de decisiones operativas y estratégicas de una organización para cumplir con los objetivos de negocio.

Business intelligence es el conjunto de metodologías, procesos, arquitecturas y tecnologías para la transformación de los datos de una organización en información, y la información en conocimiento, con el objetivo de facilitar la toma de decisiones.

El concepto de BI, de la misma forma que los conceptos de ERP, CRM y BPM, se apoya sobre algún tipo de solución tecnológica. Estas soluciones se alimentan de los datos de las diferentes bases de datos que puedan existir en una organización (por ejemplo, las bases de datos de un ERP o un CRM), así como de bases de datos públicas de otras organizaciones. Los datos de las diferentes fuentes se concentran en la base de datos propia de la solución de BI para poder ser explotados. La explotación de estos datos permite generar tres tipos de resultado (figura 20):

- **Minería de datos (*data mining*)** : Los datos pueden ser explotados sin ningún tipo de agrupación o clasificación previa a partir de técnicas analíticas y estadísticas, con la finalidad de detectar patrones entre determinados conceptos de información (por ejemplo, identificar patrones de comportamiento de los clientes de una organización en base a los datos del CRM).

- **Generación de informes (*reports*)** : Los datos pueden utilizarse de manera más compactada para la generación de informes con información de negocio más o menos complejos, con información estática o dinámica, y destinados a perfiles de usuario específicos de la organización (directivos, técnicos, etc.).

- **Cuadros de mando (*balanced scorecards*)** : Los datos pueden ser explotados de manera muy compactada para visualizar solo unos pocos indica-

dores clave de la organización, con la finalidad de poder tomar decisiones operativas a corto plazo y decisiones estratégicas a medio y largo plazo.

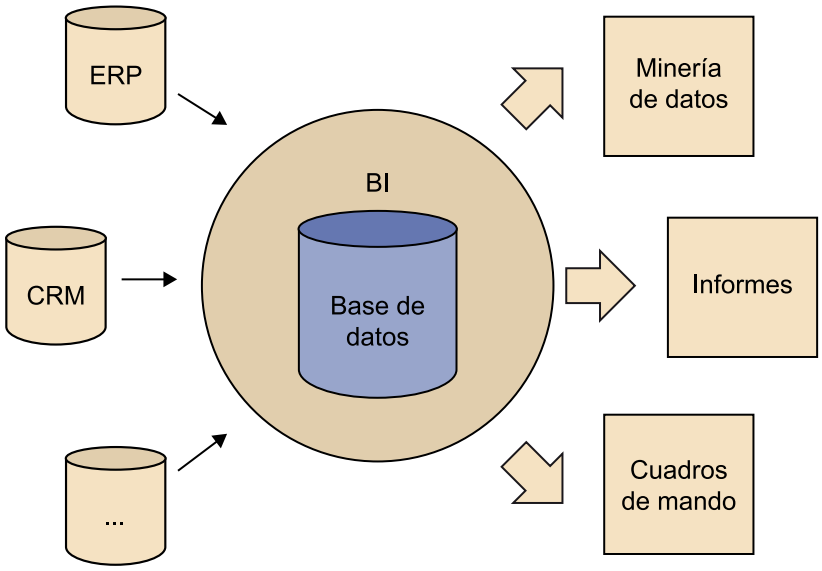

Figura 20. BI

Los datos utilizados por la solución de BI para generar conocimiento para la organización son exactamente los mismos, ya sea para aplicar minería de datos (datos completamente desagregados) o para generar cuadros de mando (datos muy agregados). Los datos serán exactamente los mismos en los dos casos y la diferencia la establecerán las técnicas y herramientas específicas utilizadas para agrupar y extraer resultados.

Una característica importante de las soluciones de BI es que los datos sobre los que aplicar "inteligencia de negocio" deben ser datos históricos (acumulados durante meses o años) y cuya integridad sea alta (datos consistentes y sin errores de formato). Ello es necesario para, después del análisis de la información, poder obtener resultados y llegar a conclusiones que sean fiables. Si no fuera así, no sería posible utilizar un sistema de BI para cumplir con su principal objetivo, que es facilitar la toma de unas decisiones que condicionarán la estrategia de la organización a medio y largo plazo.

Con el auge de Internet y las páginas web han aparecido muchas herramientas de análisis web o *web analytics* (por ejemplo, Google Analytics). Aunque la filosofía de estas herramientas puede ser semejante al *business intelligence*, hay una diferencia básica que las distingue: al contrario que las soluciones de BI (que usa datos históricos y con alta integridad), las herramientas de *web analytics* pretenden medir la actividad de los usuarios en una web durante periodos relativamente cortos de tiempo usando datos cuya integridad puede ser discutible, complementados con datos estadísticos (con margen de error). El objetivo de *web analytics* no es disponer de información fiable sobre la que tomar decisiones estratégicas, sino simplemente tener una visión aproximada de cuál es el uso que se está haciendo de la web para analizar tendencias de sus usuarios.

Algunas de las soluciones de BI más usadas son:

- Microstrategy (www.microstrategy.es/)
- Oracle BI (www.oracle.com/us/solutions/business-analytics/business-intelligence/overview/index.html)
- IBM Cognos (www-01.ibm.com/software/analytics/cognos/)
- SAP BusinessObjects (www54.sap.com/solutions/analytics/business-intelligence.html)
- Microsoft BI (www.microsoft.com/en-us/bi/default.aspx)

5.5. ECM (*enterprise content management*)

El concepto de *enterprise content management* se refiere a la gestión de todo aquello que tenga que ver con contenidos y documentos (correos electrónicos, audios, vídeos, documentos de archivo, portales de colaboración, etc.) dentro de una organización. Si el ERP se orienta a la organización, el CRM al cliente, y el BPM a procesos, ECM se orienta a contenidos (*content-oriented*).

ECM son las estrategias, los métodos y las herramientas usadas para capturar, gestionar, almacenar, preservar y distribuir contenido y documentos (independientemente de dónde se encuentren) relacionados con los procesos de negocio de una organización.

La filosofía de ECM va más allá de la simple organización y estructuración de los contenidos de una organización para que, en un momento dado, sea fácil encontrar un documento o generar una nueva versión de un documento existente. ECM pretende gestionar los contenidos de una organización para que sean útiles para alcanzar los objetivos de negocio. Para ello, es necesario gestionar todo el ciclo de vida de los documentos y los contenidos, desde el momento en que son capturados y entran en la organización hasta el momento en que son destruidos. Al igual que en los sistemas y herramientas comentados en los puntos anteriores, la filosofía de ECM se apoyará siempre en una solución tecnológica que será necesaria, pero no suficiente, para alcanzar las metas establecidas dentro de la gestión de contenidos de la organización.

Las soluciones de ECM han experimentado un importante auge desde que las organizaciones empezaron a utilizar de forma habitual contenidos y documentos electrónicos. El formato electrónico presenta un nivel de complejidad adicional al que puede presentar el formato papel, y por ello su gestión requiere de herramientas más especializadas. Además, la mayoría de soluciones de ECM permiten no solo la gestión del ciclo de vida de documentos electrónicos, sino también la gestión del ciclo de vida de documentos en papel. Indepen-

dientemente del formato utilizado, los parámetros a garantizar para cualquier contenido y documento son su autenticidad, su fiabilidad, su integridad, y su disponibilidad.

Habitualmente, se utilizan cuatro elementos para justificar la necesidad de la gestión de contenidos, y consecuentemente, de soluciones de ECM, dentro de una organización:

- **Conformidad** (*compliance*). El cumplimiento de las normativas y leyes que puedan aplicar sobre los contenidos de una organización tiene un coste en recursos (humanos y materiales), aunque puede no aportar ningún valor añadido a la misma. No obstante, el no cumplimiento de las normativas y leyes establecidas puede representar un coste mucho mayor para la organización (tanto económico como social). Es evidente que la conformidad no es un problema tecnológico, pero la tecnología deberá contribuir a limitar los riesgos de la no conformidad, especialmente desde el momento en que las organizaciones gestionan cada vez más volumen de datos no estructurados.

  Un ejemplo claro es una administración pública: asegurar que la información de la ciudadanía es capturada, almacenada, gestionada y utilizada sin salirse del marco legal establecido no es una cuestión menor (pensemos en la gestión de documentos de archivo que contienen datos sensibles acerca de un ciudadano).

- **Colaboración**. En cualquier organización, el trabajo en equipo es fundamental. El trabajo en equipo implica la interacción de varias personas o aplicaciones que pueden compartir contenidos y documentos (a través del correo electrónico, de mensajería instantánea, de un repositorio de datos común, etc.) para realizar sus tareas. Para poder hacerlo, serán necesarias metodologías y herramientas que lo permitan, pero sobre todo, serán necesarias metodologías y herramientas que garanticen que la colaboración está conforme al marco normativo y/o legal establecido.

  Por ejemplo, en algunas organizaciones es obligado guardar un registro de todas las comunicaciones que se han generado entre sus empleados o entre sus empleados y clientes.

- **Continuidad**. Imaginemos qué ocurriría si en nuestra organización desaparecen de golpe (por una catástrofe natural o por un error humano) todos los contenidos (documentos, documentos de archivo, cuentas de correo electrónico, etc.). Todo depende de la criticidad que los contenidos representen para la organización, pero para muchas de estas significaría un gravísimo problema. La correcta aplicación de estrategias y soluciones de ECM puede garantizar que ante un evento de estas características el impacto sobre la organización será nulo o mínimo y podrá continuar operando normalmente.

  Ejemplos de estrategias para garantizar la continuidad son copias de seguridad, repositorios centralizados externos a la organización, separar repositorios geográficamente, etc.

- **Costes**. ¿Cuál es el coste de no gestionar adecuadamente los contenidos de una organización? Es difícil de decir y de medir. Pero podemos hacer una aproximación. Por un lado, ya hemos apuntado el coste que pueden tener las no conformidades y la no continuidad. Por otro lado, una correcta gestión de contenidos hará a la organización más eficiente (o, como mínimo, esto es lo que se pretende) y ello repercutirá en, por ejemplo, un ahorro en el tiempo que los empleados deben invertir en recuperar, consultar y usar contenidos. Por lo tanto, el incremento de eficiencia proporcionado por ECM permitirá disminuir los costes de la organización.

Una de las características de las soluciones de ECM es, al igual que en los ERP, su modularidad. Una solución de ECM suele constar de un módulo base sobre el que se pueden ir añadiendo módulos adicionales en función de las necesidades de gestión de contenidos de la organización. Algunas de las principales funcionalidades o módulos que podemos encontrar en una solución de ECM para la gestión de contenidos y documentos electrónicos son los siguientes (figura 21):

- **Captura de contenidos**. Los contenidos pueden entrar en una organización a través de diversas fuentes, en forma de, por ejemplo, documentos en papel, formularios electrónicos o contenidos electrónicos no estructurados (correos electrónicos, mensajería instantánea, documentos de texto, hojas de cálculo, vídeos, audios, etc.). En este apartado se incluyen herramientas de:
  - **Digitalización** de documentos para transformar documentos en formato papel a una imagen en formato electrónico mediante un escáner.
  - **Procesamiento de formularios electrónicos**, especialmente de aquellos que presentan una estructura definida (se conoce la posición de cada uno de los datos dentro del formulario).
  - **Reconocimiento de caracteres** para transformar automáticamente datos manuscritos o contenidos electrónicos no estructurados a datos electrónicos mediante técnicas de reconocimiento óptico de caracteres (OCR) o de reconocimiento inteligente de caracteres (ICR).

- **Clasificación de contenidos**. Una vez capturado, el contenido debe ser clasificado según una determinada estructura (que se usará posteriormente cuando se desee consultar el contenido). Este paso puede ser llevado a cabo por personas, pero existen soluciones de ECM que son capaces de identificar rasgos característicos del contenido capturado (por ejemplo, unas marcas específicas en un documento o unos datos electrónicos clave identificados mediante OCR) y clasificarlo automáticamente.

- **Indexación de contenidos**. La captura de un contenido genera, aparte del contenido en sí mismo, un conjunto de datos que describen el contenido (metadatos), como por ejemplo, el tipo de documento, la fecha de captu-

ra, el idioma usado, etc. Las soluciones de ECM pueden permitir indexar (generar un índice) tanto el propio contenido, como sus metadatos para que las posteriores búsquedas sean rápidas.

- **Almacenamiento de contenidos**. Los contenidos capturados, clasificados e indexados son almacenados en el repositorio (o repositorios) de la solución de ECM. El repositorio de contenidos es el corazón de un ECM y, habitualmente, tiene forma de base de datos. Una vez almacenados, los contenidos pueden ser explotados por la organización.

- **Motor de búsqueda**. Probablemente una de las funcionalidades más importantes de cualquier herramienta de gestión de contenidos sea el motor de búsqueda. De nada nos servirá tener muchos contenidos si no somos capaces de encontrarlos rápidamente cuando los necesitemos. Una correcta clasificación, indexación y gestión de los repositorios de contenidos es imprescindible para asegurar el éxito de las búsquedas. Normalmente se ofrecen funcionalidades de búsqueda tanto básicas como avanzadas, y tanto por contenido como por metadatos. Un motor de búsqueda puede tener cargado todo lo que convenga, incluso un tesauro, todo depende de cómo se quiera implementar la solución.

- **Gestión de documentos (*document management*, DM)** . Se ofrecen funcionalidades básicas para la gestión de documentos, como la creación, búsqueda, consulta, edición, bloqueo (*check-out*), desbloqueo (*check-in*), control de cambios o versionado de documentos. También se ofrecen funcionalidades más avanzadas, como la agrupación o enlace de documentos (por ejemplo, para formar expedientes) o la asignación de un ciclo de vida (fases o estados por los que debe pasar) a un documento.

- **Gestión de documentos de archivo (*record management*, RM)** . Se ofrecen funcionalidades para la gestión de los documentos de archivo como la definición de calendarios de retención (cuánto tiempo debe ser guardado) y políticas de disposición (qué hacer con el documento de archivo cuando haya transcurrido el período de retención). Los documentos de archivo son las evidencias de la actividad de una organización y es por ello por lo que su gestión es de vital importancia para ella (especialmente si hablamos de organismos y administraciones públicas donde las exigencias legales en este aspecto son muy estrictas).

- **Gestión de correos electrónicos**. Se ofrecen funcionalidades para clasificar, almacenar y destruir correos electrónicos de manera consistente con las normas o los estándares que pueda seguir una organización. Estas funcionalidades suelen estar integradas con las funcionalidades de gestión de documentos de archivo. Dado el gran volumen de correos electrónicos que se genera en las organizaciones actuales, el poder gestionar un correo elec-

trónico como cualquier otro documento de la organización resulta muy interesante.

- **Gestión de contenidos web**. Se ofrecen funcionalidades específicas para la creación, revisión, aprobación y publicación de contenidos web.

- **Gestión de activos digitales**. De la misma forma que se gestionan documentos, se pueden gestionar documentos multimedia, como por ejemplo, fotografías, animaciones, vídeos, música, o incluso código software.

- **Publicación de contenidos**. Se ofrecen funcionalidades para hacer llegar los contenidos a quien los pueda necesitar, ya sea, por ejemplo, vía impresión, correo electrónico, portales de colaboración (intranet), páginas web (Internet), mensajes de texto, o RSS (*really simple syndication*).

- **Federación de contenidos**. Se ofrecen funcionalidades para poder realizar búsquedas y recuperar contenidos que están en repositorios externos a la organización (en otra plataforma de ECM). Para ello, se puede utilizar el protocolo estándar CMIS (*content management interoperability services*) de OASIS.

- **Conversión de formato de los contenidos**. Se ofrecen funcionalidades para migrar contenidos en formatos obsoletos a formatos actuales. Es habitual migrar documentos ofimáticos de sus formatos de origen a formatos que se consideran más duraderos en el tiempo como PDF o XML.

- **Copias de seguridad de los contenidos** (*backup*). Se ofrecen funcionalidades para realizar copias de seguridad de los contenidos y evitar la pérdida de información debida a fallos de la tecnología, errores humanos o desastres naturales.

- **Gestión de la seguridad de los contenidos**. Se ofrecen funcionalidades para controlar y restringir el acceso a los contenidos durante su creación, gestión y publicación. Algunas de las herramientas usadas habitualmente para garantizar la seguridad de los contenidos son herramientas de autenticación (usuario y contraseña), firmas electrónicas (para verificar la autenticidad de contenidos firmados electrónicamente), encriptación de repositorios (encriptación de contenidos), protocolos de comunicación segura (encriptación de datos durante su transporte a través de una red) o herramientas de gestión de derechos digitales (*digital rights management* o DRM). Además, todo ello se puede complementar con el registro de quién o cuándo ha accedido a un determinado contenido del repositorio.

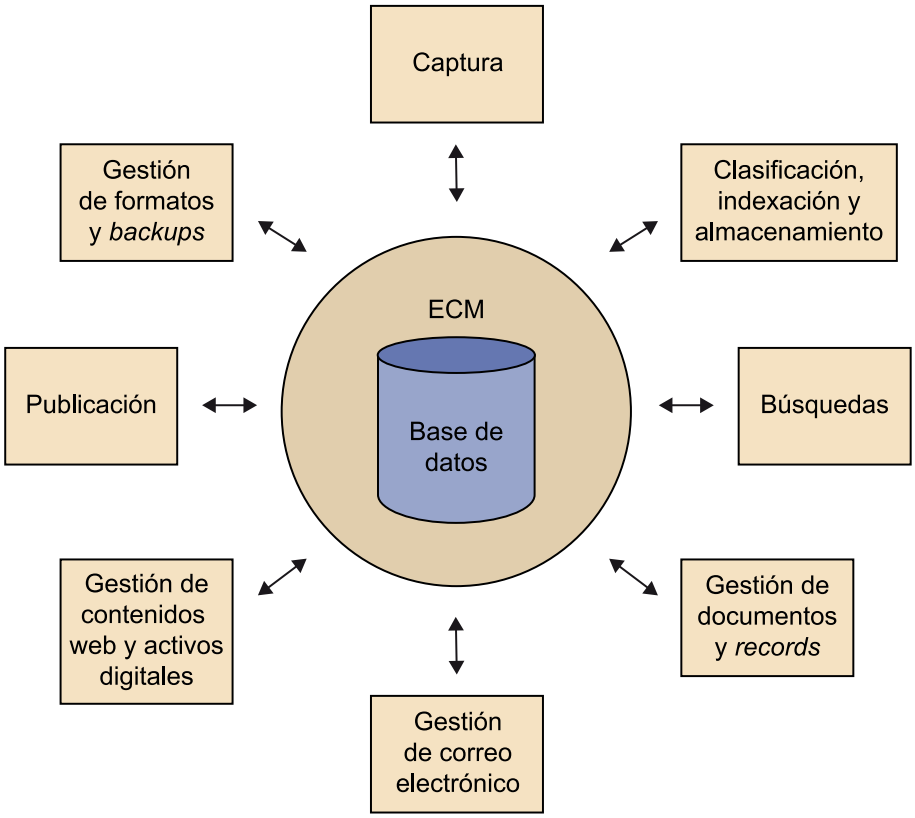

Figura 21. ECM

Un concepto importante a tener en cuenta cuando hablamos de ECM es el concepto de ciclo de vida de un contenido o un documento. El ciclo de vida es el conjunto de estados (por ejemplo, "borrador", "pendiente de aprobación", "aprobado", "obsoleto", etc.) en los que se puede encontrar un contenido o un documento desde su creación hasta su eliminación, así como las reglas que deben aplicarse (por ejemplo, cambio de los permisos asociados al contenido o documento) y condiciones que deben satisfacerse (por ejemplo, verificación del valor de un metadato) para pasar de un estado a otro. La mayoría de soluciones de ECM incorporan herramientas de automatización del ciclo de vida (llamadas herramientas de gestión del flujo de trabajo o herramientas de *workflow*), es decir, herramientas para controlar quién, cómo y cuándo se debe realizar un determinado cambio de estado dentro de un proceso de gestión de un contenido o documento (salvando las distancias, podríamos considerar las herramientas de *workflow* como herramientas de BPM muy básicas y con un alcance muy acotado).

Otro concepto a destacar es la capacidad de una solución de ECM de integrarse con otros sistemas y herramientas en una organización o fuera de ella. La mayoría de procesos de negocio que podemos encontrar dentro de una organización pasan en algún momento por la creación o uso de un contenido o documento.

Por ejemplo, desde nuestro CRM enviamos un correo electrónico a un cliente, o una aplicación específica ha generado un nuevo expediente electrónico que debemos guardar en el repositorio del ECM.

Una solución de ECM ha de ser flexible para adaptarse a la organización y capaz de mostrar sus capacidades a los demás sistemas para que estos puedan hacer uso de ellas. Para ello, las soluciones de ECM suelen seguir tres tipos de estrategia de integración (que no son excluyentes unas con otras) (figura 22):

- **Arquitectura SOA**: Muchas de las soluciones actuales (ya sea de origen o porque han evolucionado) presentan arquitecturas orientadas a servicios de manera que ofrecen sus funcionalidades en forma de servicios a otros sistemas de información.

- **Conectores**: Algunas de las soluciones ofrecen conectores específicos para integrarse con sistemas de información concretos.

  Por ejemplo, son habituales los conectores con los sistemas de correo electrónico como Microsoft Exchange o con sistemas ERP de uso muy extendido como SAP.

- **Interfaces de programación de aplicaciones**: La mayoría de soluciones proporcionan a los desarrolladores de sistemas de información interfaces de programación de aplicaciones (en inglés, *application programming interface* o API) que ofrecen la posibilidad de desarrollar nuevas funcionalidades en base a las que ya ofrece la solución de ECM de forma nativa.

API

API (*application programming interface*) es el conjunto de funciones y procedimientos que ofrece un sistema para que otros sistemas las usen de base para desarrollar nuevas funcionalidades.

Una integración casi obligada de una solución de ECM es la integración con una solución de BPM. A través de una solución de BPM podremos relacionar los procesos de gestión del negocio con la gestión de contenidos y documentos de la organización. De esta manera, los ciclos de vida de contenidos y documentos (con sus respectivos *workflows*), definidos como parte de los procesos de gestión documental en la solución de ECM, pueden ser gestionados conjuntamente con el resto de procesos de negocio de la organización. La integración entre ECM y BPM se puede realizar a través de servicios SOA, a través de API, a través de conectores específicos o, lo que ofrecen algunas grandes compañías tecnológicas como IBM o EMC, a través de plataformas de soluciones, que incluyen de forma nativa una solución de ECM integrada con una solución de BPM.

En el mercado podemos encontrar tanto soluciones de ECM completas (incluyen la mayoría de módulos) como soluciones puntuales (incluyen un módulo o funcionalidad determinada normalmente con capacidad de integrarse con otras soluciones de ECM). Algunas de las plataformas de ECM más completas son:

- IBM Filenet (www.ibm.com/software/ecm/filenet)

- EMC Documentum (www.emc.com/domains/documentum/index.htm)

- OpenText (www.opentext.com/2/global/products/enterprise-content-management.htm)

Reflexión

Las plataformas de ECM más completas que podemos encontrar en el mercado pertenecen a compañías de origen anglosajón (norteamericanas, como IBM o EMC, o canadienses, como OpenText). Esto tiene muchas ventajas: son grandes compañías con sede central en países estables, que llevan muchos años en el mercado y sus servicios de soporte al usuario son de gran calidad y a nivel mundial; pero también tiene un inconveniente: la tradición anglosajona en lo que a gestión de documentos se refiere no es exactamente la misma que la que podemos aplicar en la mayoría de países europeos. Y ello se traduce en soluciones de ECM que, a pesar de cubrir la gran mayoría de casuísticas, no contemplan o enfocan de manera diferente a como estamos acostumbrados algunas problemáticas concretas de gestión documental.

Un par de ejemplos son la firma electrónica y el concepto de "expediente" (agrupación de documentos con un ciclo de vida propio). El uso de la firma electrónica y la gestión de "expedientes" (y toda la complejidad que esta gestión conlleva, especialmente cuando entramos en la gestión de documentos de archivo) es una práctica común a nivel europeo, pero muy poco usada en el mundo anglosajón. Las soluciones de ECM procedentes de este mundo no contemplan (o lo hacen a nivel muy genérico) ni el uso de firmas electrónicas ni la gestión de "expedientes" (aunque se empiezan a incorporar soluciones genéricas bajo el título de *case management*), de manera que obligan a realizar desarrollos específicos (con su coste asociado) para personalizarlas.

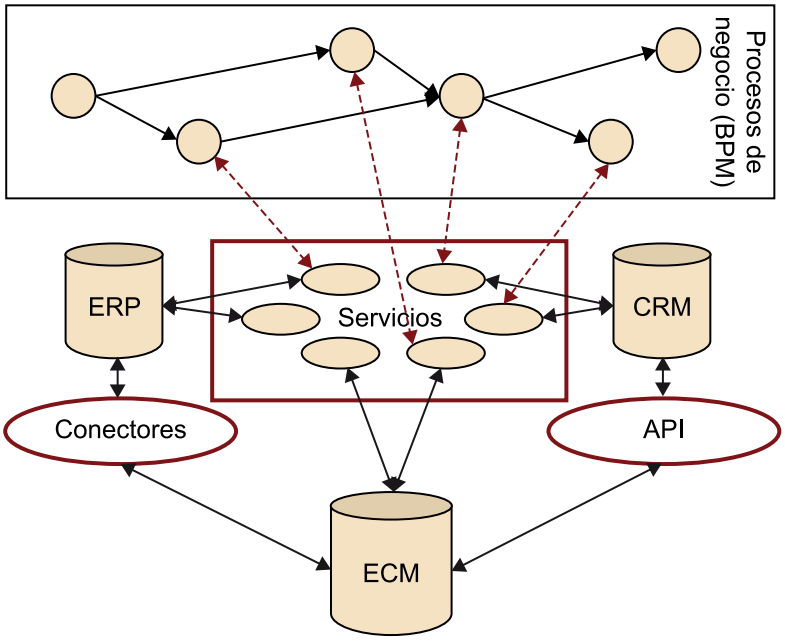

Figura 22. Ejemplo de integración de sistemas de gestión de datos

Actividades

Identificad el tipo de arquitectura y los diferentes componentes de los sistemas de información con los que el estudiante trabaja habitualmente.

Bibliografía

Arquitectura de sistemas

ATOS. Plataforma de arquitectura y aplicaciones. Disponible en: http://es.atos.net/es-es/sobre_nosotros/filosofia-de-empresa-e-innovacion/thought_leadership/container/wp_plataforma_arquitectura_aplicaciones.htm

Rozanski, N.; Woods, E. (2011). "Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives". *Addison-Wesley Educational Publishers Inc*. (2.ª ed., parte I, cap. 2, pág. 11-29).

Marquès i Puig, J. M.; Navarro Moldes, L. (coords.); García López, P. A. (2007). Arquitectura de sistemas distribuidos (recurso electrónico). Barcelona: Universitat Oberta de Catalunya. ISBN 8497074416; Código módulo XP07/81068/00079

SOA

ATOS. SOA. Disponible en: http://es.atos.net/es-es/sobre_nosotros/filosofia-de-empresa-e-innovacion/thought_leadership/container/wp_soa.htm

Josuttis, N. M. (2007). *SOA in practice*. Nicolai Josuttis; Sebastopol, CA: O'Reilly. ISBN 9780596529550

Bases de datos

Berni Millet, P.; Gil de la Iglesia, D. (2010). *Diseño de bases de datos*. Barcelona: UOC, Universitat Oberta de Catalunya. ISBN 9788469294307

Silberschatz, A.; Korth, H. F.; Sudarshan, S. (2002). *Fundamentos de bases de datos* (4.ª ed.). Madrid: McGraw-Hill.

Traducción: Fernando Sáenz Pérez y otros. Revisión técnica: Luis Grau Fernández. ISBN 8448136543

Ramez, E.; Shamkant B. Navathe (2002). *Fundamentos de sistemas de bases de datos* (3.ª ed.). Madrid: Addison Wesley.

Traducción: Verónica Canivell Castillo, Beatriz Galán Espiga, Gloria Zaballa Pérez. Revisión técnica: Alfredo Goñi Sarriguren, Arturo Jaime Elizondo, Tomás A. Pérez Fernández. ISBN 8478290516

Silberschatz, A.; Korth, H. F.; Sudarshan, S. (2006). *Fundamentos de bases de datos* (5.ª ed.). Madrid: McGraw-Hill.

Traducción: Fernando Sáenz Pérez y otros. Revisión técnica: Luis Grau Fernández. ISBN 8448146441

Motores de búsqueda

Leloup, C. (1998). *Motores de búsqueda e indexación*. Barcelona: Gestión 2000. ISBN 8480882573

ERP

Muñiz González, L. (2004). *ERP: guía práctica para la selección e implantación* (ERP: enterprise resource planning, sistema de planificación de recursos empresariales). Barcelona: Gestión 2000. ISBN 8480883596

CRM

ATOS. CRM. Disponible en: http://es.atos.net/es-es/sobre_nosotros/filosofia-de-empresa-e-innovacion/thought_leadership/container/wp_customer_relationship_management_crm.htm

Greenberg, P. (2003). *Las Claves de CRM: gestión de relaciones con los clientes*. Madrid: McGraw-Hill. ISBN 8448138007

BPM

ATOS. Business Process Management. Disponible en: http://es.atos.net/es-es/sobre_nosotros/filosofia-de-empresa-e-innovacion/thought_leadership/container/wp_Business_Process_Management_BPM.htm

Cummins, F. y otros (c 2009). *Building the agile enterprise* (recurso electrónico). Amsterdam / Boston: MK/OMG Press/Elsevier. ISBN 9780123744456

ECM

White, Martin S. (2005). *The Content management handbook*. Londres: Facet Publishing. ISBN 1856045331

Addey, D. y otros (2002). *Content management systems*. Arden House: Glasshaus. ISBN 190415106X