



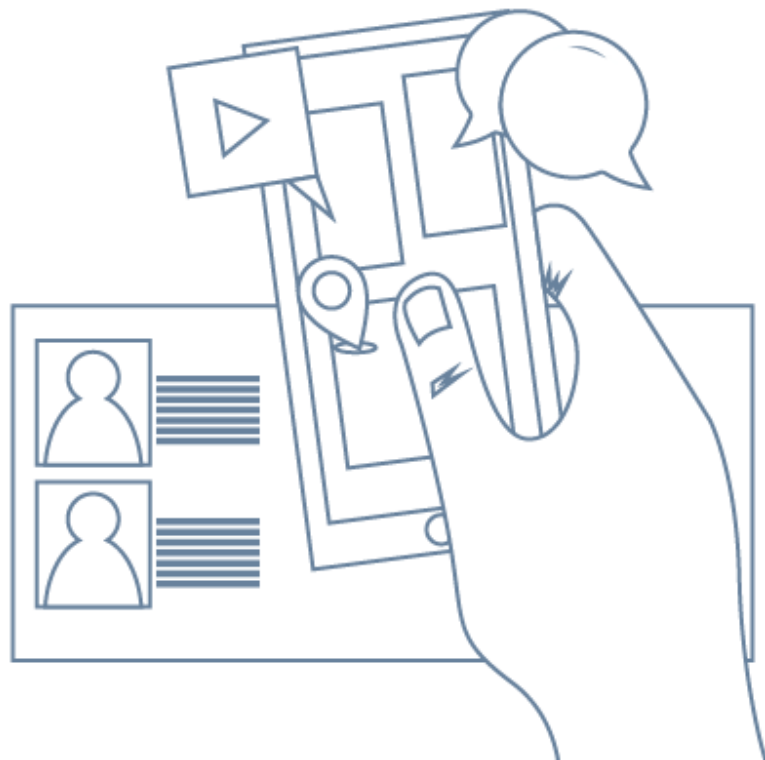
BASES DE DATOS RELACIONALES

MODELAMIENTO DE BASES DE DATOS RELACIONALES



BASES DE DATOS RELACIONALES

MODELAMIENTO DE BASES DE DATOS RELACIONALES





ESCUELA DE CONSTRUCCIÓN E INGENIERÍA

Director de Escuela / Marcelo Lucero

ELABORACIÓN

Experto disciplinar / Aida Villamar Gallardo

Diseñador instruccional / Oscar González Cantin

VALIDACIÓN

Experto disciplinar / ---

Jefa de diseño instruccional y multimedia / Alejandra San Juan Reyes


DISEÑO DOCUMENTO

Equipo de Diseño Instruccional AIEP



Contenido

APRENDIZAJES ESPERADOS	6
INTRODUCCIÓN.....	7
BASES DE DATOS.....	8
La arquitectura de una base de datos	8
¿Qué es un dato?	9
Clasificación de datos	9
Tipos de datos más importantes	10
¿Qué es un proceso?	11
Sistema de información (SI).....	11
Clasificación de los sistemas de información.....	14
Ciclo de vida de un sistema de información (SDLC).....	16
Modelos para planificar el proceso	19
Calidad dentro del desarrollo de un sistema de información	26
Garantía de la calidad del software	26
Calidad de software	27
Costo de la calidad	27
MODELO DE NEGOCIOS	28
Modelo de negocio Canvas.....	28
Segmentos de clientes	29
Propuesta de valor.....	29
Canales.....	30
Relación con el cliente	30
Fuente de ingresos.....	31
Recursos clave.....	31
Actividades clave.....	32
Socios clave.....	32
Estructura de costos	33
ARQUITECTURA DE UN SISTEMA DE BASE DE DATOS	34
Objetivos de los DBMS.....	34
Independencia lógica y física de los datos.....	35
Requerimientos de clientes	36
Definición de requerimiento.....	36



Importancia de los requerimientos	37
Documentos de requerimientos.....	39
Clasificación de requerimientos	40
Contextualización del análisis del problema (antecedentes)	43
MODELOS DE DATOS.....	44
Tipos de modelos de datos	46
Modelo conceptual.....	46
Modelo lógico	47
Modelo físico	49
REQUERIMIENTOS INICIALES Y REALES PARA LA MEJORA DE UN MODELO DE DATOS EN UNA BASE DE DATOS	50
Análisis de los requerimientos.....	50
Reconocer las necesidades del sistema.....	50
Recopilación de información	50
CONCLUSIÓN	51
IDEAS CLAVE	51
REFERENCIAS BIBLIOGRÁFICAS	53



APRENDIZAJES ESPERADOS

¿Qué aprenderás en este módulo, por unidad y por semana? Revisa la competencia asociada, los aprendizajes esperados y los criterios de evaluación de las actividades de aprendizaje que realizarás semana a semana.

Competencia del módulo:	Al finalizar tu módulo, serás capaz de realizar programas en pseudolenguaje considerando sentencias de control y operadores lógicos.	Duración:	72 horas
		Modalidad:	Online
Aprendizaje esperado de la semana	Analizan la arquitectura de una base de datos según requerimientos propuestos por el cliente.		
Criterios de evaluación de la semana	<ol style="list-style-type: none">1. Identificar arquitectura y conceptos asociados al modelamiento de una base de datos.2. Identificar requerimientos iniciales para el modelamiento de una base de datos.3. Clasificar tipos de modelos de datos para establecer una base de datos.4. Seleccionar requerimientos iniciales con los reales para la mejora de un modelo de datos en una base de datos.		



Introducción

Semana 1

¿BASES DE DATOS? ¿PARA QUÉ SIRVEN?

En el módulo de Bases de datos relacionales deberás realizar una serie de tareas particulares, con el fin de optimizar el trabajo de tus futuros clientes y garantizar el correcto funcionamiento de los datos que ellos te suministren.

¿Y cómo harás esto? Será, por supuesto, mediante una **base de datos**.

A lo largo de esta unidad, aprenderás una serie de conceptos, técnicas e instrumentos que te ayudarán a desempeñarte de manera efectiva y eficiente en un contexto profesional.

Piénsalo de esta manera: nuestro cerebro funciona como una profunda y enorme red de datos. La información recopilada en este órgano vital está guardada en distintos lugares. Cuando requerimos ejecutar alguna acción (movernos, hablar e incluso cuando sentimos algo), cierta información, y no otra, es desempaquetada para ayudarnos a realizar lo que queremos.

Del mismo modo, una base de datos posee características que un ser humano por sí solo no podría replicar.

Te invitamos a aprender, descubrir y crear nuevas formas de ver el mundo. Verás que una base de datos puede ser algo más que mera información.

BASES DE DATOS

La arquitectura de una base de datos

Antes de revisar lo concerniente a la arquitectura de una base de datos, estudiarás algunos conceptos básicos para contextualizar y tener una visión general. Luego, revisaremos los contenidos de esta semana en detalle.



¿Qué es una base de datos?

Una **base de datos** es una colección de información organizada de forma que los programas que están en los computadores puedan tomar rápidamente aquellos fragmentos de datos que necesiten para su funcionamiento.

El concepto de **modelado de datos** está asociado a la forma de establecer una estructura y organización de los datos para que puedan ser accedidos fácilmente por las bases de datos, esto por medio de un análisis de la situación actual en el entorno en que se desea implementar.

En términos generales, los modelos de datos son “una colección de herramientas conceptuales, reglas y convenciones utilizadas para describir los datos, sus relaciones, sus dominios y las restricciones de integridad” (Martínez y Gallegos, 2017).

Hay tres características importantes inherentes a los sistemas de bases de datos:

- La separación entre los programas de aplicación y los datos.
- El manejo de múltiples vistas por parte de los usuarios.
- El uso de un catálogo para almacenar el esquema de la base de datos.

Asimismo, Martínez y Gallegos destacan que “Para poder cumplir estas tres importantes características, el comité ANSI-SPARC propuso, en 1975, una arquitectura de tres niveles para los DBMS, dividiéndola en los niveles de abstracción interno, conceptual y externo” (2017).

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. Los niveles de abstracción se dividen en:

- Nivel interno o físico.
- Nivel conceptual o lógico.
- Nivel externo.

Un ejemplo de arquitectura de datos bien definida son aquellas empresas que, tras conocer los cambios que se iban a aplicar a la RGPD (Reglamento General de Protección de Datos) en España, adaptaron sus bases de datos antes de que la normativa entrara en vigor.

¿Qué es un dato?

Un dato es una **representación simbólica** (numérica, alfabética, algorítmica, etc.) de un **atributo** de una **entidad**. Es preciso destacar que un dato no tiene sentido o significado por sí mismo, sino que sirve para, por ejemplo, realizar cálculos o tomar ciertas decisiones. Los datos pueden ser comprimidos, encriptados, transmitidos y almacenados.



En su nivel más básico, los datos de un computador corresponden a infinitas combinaciones entre ceros y unos, conocidos como **datos binarios**.

Por otra parte, el concepto **entidad** refiere a un objeto que existe en el mundo real y que se diferencia de otros objetos. Así, cada uno de nosotros podría ser una entidad con determinados **atributos**, o las características particulares de cada uno. Así, las entidades se describen en una base de datos a través de un conjunto de atributos.

Datos v/s información



¿Cuál es la diferencia entre dato e información? Generalmente, estos dos conceptos se utilizan indistintamente, pero son muy diferentes.

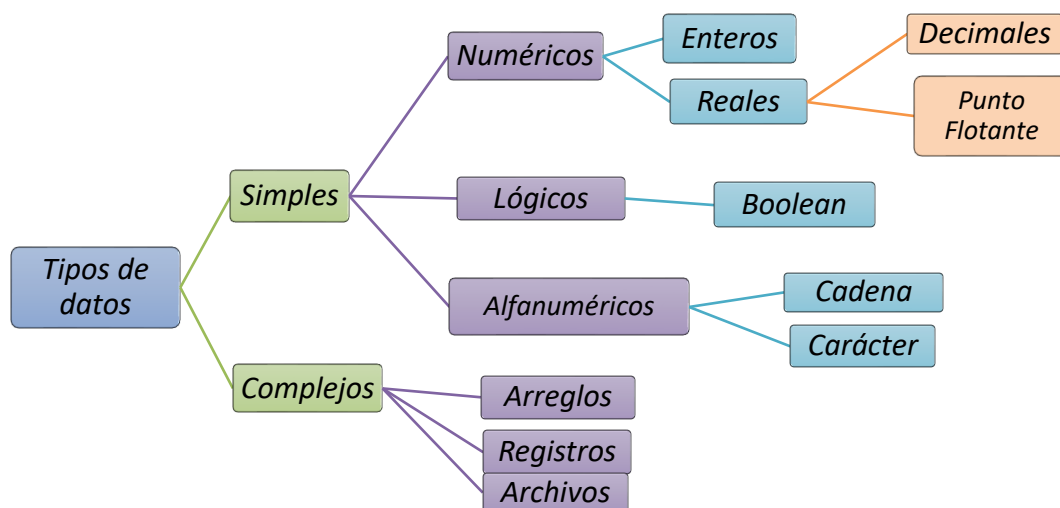
La información corresponde al sentido que se le da a un conjunto de datos. Por ejemplo: tenemos los números 16, 17, 17, 17, 17, 15, 15. Se convierten en información cuando se sabe que representan las edades de un grupo de alumnos de un curso.

Clasificación de datos

Podemos clasificar los datos de acuerdo a su estructura. Existen datos simples y datos compuestos:

- Los **datos simples** son indivisibles. En el caso de que quisiéramos dividirlos, estos dejarían de considerarse datos y pasarían a ser meras partes o divisiones del dato. Estas partes del dato no poseen ningún significado.
- Los **datos compuestos** son un conjunto de datos simples, los que colaboran en la estructuración y sentido del dato compuesto dentro de un modelo de datos. Esta composición se produce cuando los datos son capturados desde el medio donde se encuentran. También se pueden componer o combinar cuando ya están capturados.

Revisa a continuación los tipos de datos según estructura:



Tipos de Datos. Fuente: elaboración propia.

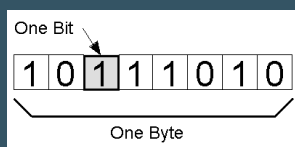
Tipos de datos más importantes

Un tipo de dato corresponde al tipo de valor que puede contener una columna. Por ejemplo, si contendrá datos enteros, o tal vez datos de caracteres, datos de fecha y hora, cadenas binarias, entre otros.

En una tabla de base de datos, es necesario que cada columna cuente con un nombre y posea un tipo de datos.

Antes de continuar, es necesario que definamos uno de los conceptos que te ayudarán a comprender los tipos de datos: el **byte**.

¿Qué es un byte?



Un byte es un conjunto de 8 bits. Es considerado como una unidad y constituye el mínimo elemento de memoria direccionable de una computadora.

Por ejemplo, 4 bytes = 32 bits.

Importante: revisa la tabla de tipos de datos, ubicada como documento Anexo en el Contenido Obligatorio de la Semana.

¿Qué es un proceso?

Un proceso es la ejecución por parte del microprocesador de diversas instrucciones (actividades, acciones y tareas), de acuerdo con lo que indica un programa.

Un proceso puede, de manera informal, pensarse como un programa en ejecución. Se define como "una unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, posee un estado actual y utiliza un conjunto de recursos de sistemas asociados" (Stallings, 2005).

Recuerda que...



El término *información* refiere a un conjunto de datos organizados y procesados que componen mensajes, instrucciones, operaciones, funciones y cualquier tipo de actividad que tenga lugar en relación con un computador. La información está constituida por un grupo de datos ya supervisados y ordenados, que sirven para construir un mensaje. Un autor como Czinkota, en su libro *Marketing internacional*, declara: "la información consiste en un conjunto de datos que han sido clasificados y ordenados con un propósito determinado" (2008).

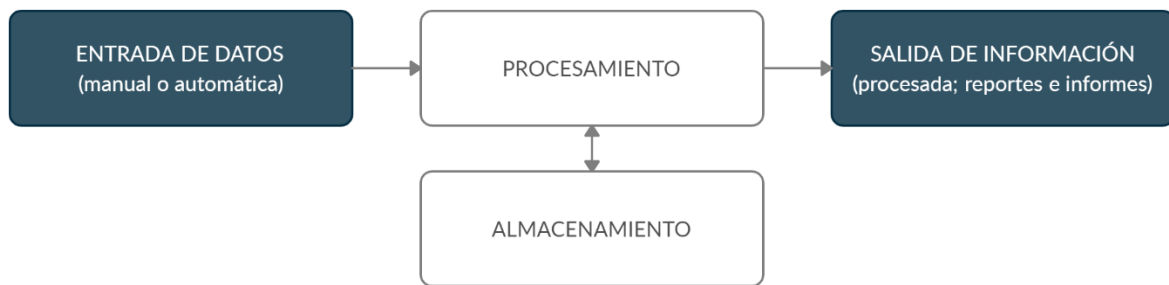
Sistema de información (SI)

Según Alejandro Peña Ayala, en su libro *Ingeniería de Software para crear sistemas de información*, "un sistema de información es un conjunto de elementos relacionados cuya finalidad es atender las demandas y necesidades de información de una organización" (2006).

Algunos ejemplos de sistemas de información se relacionan con:

- **Los sistemas de control de calidad.** En este caso, se solicita retroalimentación al cliente con el fin de evaluar los resultados de forma estadística, para, finalmente, elaborar un informe que sea interpretable por la gerencia de una determinada empresa.
- **Las bases de datos de una biblioteca.** Con esto nos referimos al lugar donde están almacenados los documentos de la biblioteca. Así, el sistema tendrá la misión de localizar y recuperar cada tomo, revista o papel de forma rápida y eficiente.
- **Las hojas de cálculo.** La planilla de toda la vida, en la que ingresamos información para organizarla de manera lógica y/o cuantificable.

Revisa, a continuación, las actividades básicas de un sistema de información y la interacción de un proceso dentro de un sistema de información:



Actividades básicas de un sistema de información. Fuente: elaboración propia.

A continuación, estudiaremos la descripción de las principales subetapas de la etapa de **procesamiento** dentro de un sistema de información:



Fuente: elaboración propia



Antes de continuar, es preciso destacar algunos puntos relacionados con el almacenamiento de los datos y su organización.

Ya aprendimos que los datos almacenados se organizan de cierta manera. Veamos cuáles son las características de cada una de estas partes.

- **Campo:** corresponde a un grupo de caracteres que identifican a un determinado elemento (persona, lugar, etc.) Por ejemplo, en la tabla ubicada más abajo se pueden apreciar campos como Expediente, Nombre o Código Postal, entre otros.
- **Registro:** es un conjunto de campos que se relacionan entre sí. Por ejemplo, la nómina de empleados podría componerse por el nombre, apellido, dirección y sueldo. En este caso, un registro corresponde a: Código del empleado A, Apellido del empleado A, Nombre del empleado A. Otro registro sería: Código del empleado B, Apellido del empleado B, Nombre del empleado B, etc.
- **Tabla (archivo):** corresponde al conjunto de registros relacionados entre sí. Es decir, se conforma mediante un grupo de registros. Por ejemplo, una “planilla marzo 2021” podría estar compuesta por registros de la nómina de todos los trabajadores de una empresa durante el mes de marzo de 2021. En este caso, la tabla se denominaría *Empleados*.
- **Base de datos:** es el conjunto **integrado** de registros relacionados entre sí. Para continuar con el ejemplo, podríamos decir que corresponde a la base de datos de trabajadores de una empresa que incluya archivos de las planillas de todos los meses, junto con otros archivos relacionados con diversas materias (como evaluaciones de desempeño, asistencia a actividades de capacitación, etc.).

	Campos				
	Expediente	Nombre	Apellidos	Dirección	Código Postal
Registro 1	1	Pedro	Pérez Parra	Pasaje Grande 512	202051
Registro 2	2	Ana	Araneda Altamirano	Calle Angosta 1024	202155
Registro 3	3	Juan	Jiménez Jadue	Avenida Eterna 2048	201144
Registro 4	4	María	Méndez Moraga	Camino sin retorno 4096	201936

En la siguiente tabla se puede apreciar que existen cuatro registros en la tabla (destacados en verde), los campos (destacados en azul) y, finalmente, un registro como tal (destacado en rojo).

Fuente: elaboración propia.

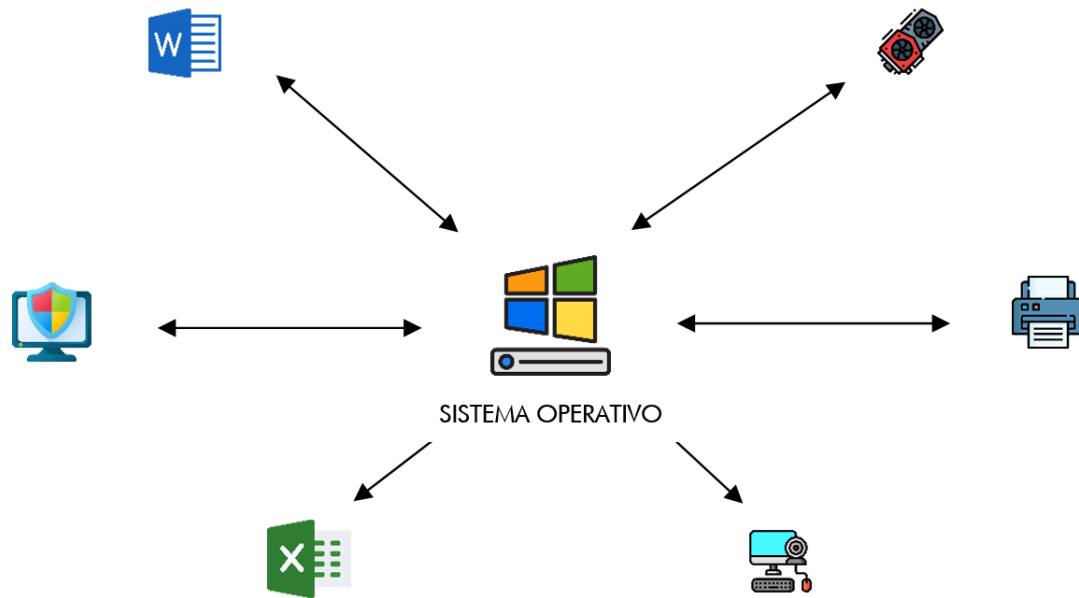
A modo de conclusión, es posible establecer que un sistema de información debe tener, necesariamente, control sobre el desempeño de la aplicación. En otras palabras, el SI tiene que generar una retroalimentación respecto a las actividades de entrada, procesamiento, almacenamiento y salida de información. Finalmente, dicha retroalimentación deberá someterse a un proceso de evaluación con el fin de determinar si el sistema cumple con los estándares de desempeño establecidos.



Clasificación de los sistemas de información

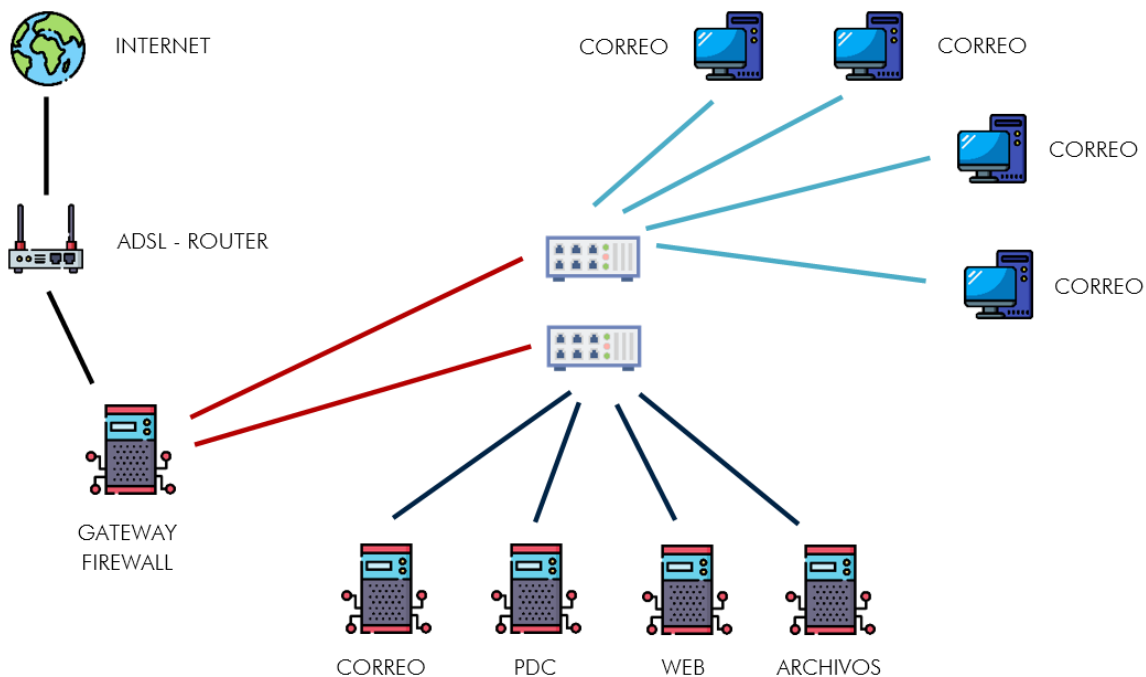
Los sistemas de información se clasifican o agrupan en tres tipos: según su propósito, estructura-funcionamiento y organización física. Revísalos a continuación:

- a) **Sistema de información según el propósito:** Pueden ser de tipo transaccional, de sistema de soporte, o sistemas estratégicos.
- **Sistemas transaccionales:** son aquellos que automatizan procesos operativos en una empresa. Su función principal radica en procesar transacciones como pagos, contratos, papeletas, planillas, etc.
 - **Sistemas de soporte:** son aquellos que se basan en modelar y analizar una gran cantidad de datos, y que permiten entregar varias alternativas para escoger. Son empleados como apoyo en la toma de decisiones.
 - **Sistemas estratégicos:** son aquellos desarrollados en las empresas con la finalidad de lograr ventajas competitivas por medio del uso de la tecnología de información.
- b) **Estructural y de funcionamiento:** pueden ser de tipo manual, mecanizado, o computacional.
- **Manual:** cuando una persona utiliza equipos (máquinas de escribir, calculadora, archivos, etc.) y realiza las principales funciones de recopilar, registrar, almacenar, calcular y generar información.
 - **Mecanizado:** cuando alguna maquinaria efectúa las funciones principales de procesamiento.
 - **Computacional:** son los sistemas que hacen uso de un computador. Hay interacción entre humano y máquina. Estos sistemas se dividen en:
 - a) **Bath:** El usuario proporciona los datos requeridos para que se ejecute un proceso y espera que el computador finalice la tarea para obtener los resultados.
 - b) **En línea:** hay una interacción directa entre ambos “participantes” (usuario y computador, en este caso) durante la ejecución de un proceso.
- c) **De acuerdo con la organización física:** pueden ser centralizados o distribuidos.
- **Centralizados:** los recursos y los procesos se encuentran centralizados en un computador central específico, por lo que su acceso se realiza por medio de terminales remotos que acceden a ellos. A continuación, se muestra una organización centralizada donde se puede apreciar un equipo central que posee el sistema operativo y los distintos terminales que acceden a él.



Sistema centralizado. Fuente: elaboración propia.

- **Distribuidos:** los recursos se encuentran ubicados físicamente en distintos lugares. Corresponde a una serie de computadores conectados mediante una red de comunicaciones. El usuario lo percibe como un solo sistema y accede a los recursos de la misma manera en que accede a los recursos locales.



Sistema distribuido. Fuente: elaboración propia.

¡Piensa rápido!



ERP (Enterprise Resource Planning)

El ERP, o sistema de planificación de recursos empresariales, se utiliza para el manejo de los recursos de una empresa. Este mezcla funciones de varias áreas en una misma aplicación. Al mismo tiempo, utiliza una base de datos para entregar información de forma más sencilla para comunicarse entre ellas. Finalmente, permite integrar, mejorar y estandarizar los distintos procesos que se realizan.

Según las clasificaciones vistas anteriormente, ¿a qué SI pertenece?

Respuesta¹

Ciclo de vida de un sistema de información (SDLC)

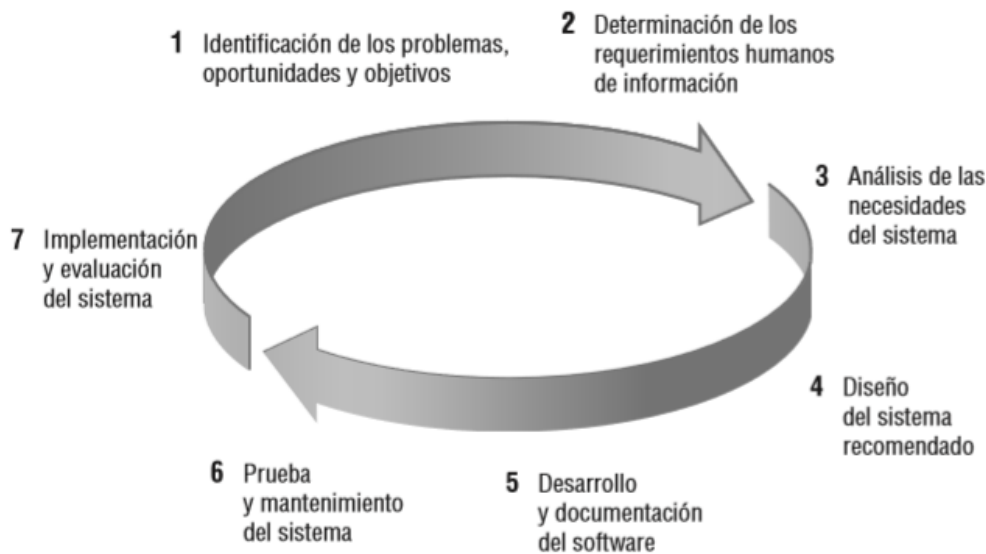
O *Systems Development Life Cycle*, en inglés. Destacaremos que todo sistema de información pasa por una serie de etapas a lo largo de su vida. Un ciclo de vida considera una serie de fases las que serán revisadas a continuación, dentro de lo que se conoce como metodología para el desarrollo de software.

Metodología para el desarrollo de software (ciclo de vida)

Este concepto refiere a un modo sistemático para administrar un proyecto, con la finalidad de terminarlo con grandes posibilidades de alcanzar el éxito. Este modo sistemático de trabajo permite dividir un proyecto de gran envergadura en módulos más pequeños (etapas). Del mismo modo, las acciones ligadas a estos módulos más pequeños colaboran para definir, por ejemplo, sus entradas y salidas. Pero, aún más importante, sistematiza el modo en que se administrará el proyecto. En resumen, una metodología para el desarrollo de software implica los procesos a seguir de manera sistemática con el fin de pensar, efectuar y mantener un producto (en este caso, un software) desde el momento en el que aparece la necesidad del software hasta que el objetivo por el que fue creado se cumple.

Así, podemos considerar que ciclo de vida de un producto tiene siete etapas bien diferenciadas:

¹ Sistemas de información estratégicos.



Las siete fases del ciclo de desarrollo de sistemas (SDLC). Fuente: Kendall y Kendall, 2011.

1. Identificación de los problemas, oportunidades y objetivos

En esta primera fase, el analista estará encargado de identificar de forma correcta tres elementos muy importantes: problemas, oportunidades y objetivos. Por lo mismo, esta etapa es importantísima para el éxito del proyecto. Asimismo, se evalúan las alternativas de solución propuestas, considerando, incluso, opciones que no consideren un sistema computacional, lo que significaría que el proyecto no continúe. Quienes participan en esta primera fase son tres: usuarios, analistas y administradores de sistemas.

En esta fase, las actividades se relacionan con entrevistar a los encargados de la administración de los usuarios, además de sintetizar los conocimientos obtenidos, estimar el alcance del proyecto y generar documentación con los resultados obtenidos.

2. Determinación de los requerimientos humanos de información

En la segunda fase, el analista deberá definir las necesidades de los usuarios involucrados mediante una serie de herramientas. El objetivo es comprender la manera en que interactúan en el contexto de trabajo con sus propios sistemas de información actuales. Algunas herramientas específicas que puede utilizar el analista son **entrevistas** o **investigación de datos**, en conjunto con **cuestionarios** y **métodos discretos**. Estos últimos consisten en, por ejemplo, analizar los procesos de toma de decisiones y el contexto de oficina. También existen **métodos integrales**, como la creación de prototipos.

En esta fase, participan usuarios y analistas. Del mismo modo, pueden participar cargos de gerencia y gente de operaciones.



3. Análisis de las necesidades del sistema

En esta tercera fase, el analista deberá examinar las necesidades del sistema. En esta fase se emplean herramientas como los diagramas de flujo de datos (DFD), diagramas de secuencia y otras técnicas que colaboran para determinar los requerimientos (más adelante lo veremos en detalle).

Durante esta etapa, el profesional debe analizar cada una de las decisiones estructuradas que se han ejecutado. Para las **decisiones estructuradas** es posible determinar condiciones, alternativas de condición, acciones y reglas de acción. En este punto del ciclo de vida, el analista debe desarrollar una propuesta de sistemas con el fin de resumir todo lo que ha recopilado acerca de los usuarios, junto con la capacidad de uso y el beneficio de los sistemas actuales. Finalmente, el analista debe realizar un análisis de costo-beneficio respecto a las distintas opciones. En caso de ser solicitado, efectúa recomendaciones. Así, en caso de que la administración acepte alguna de las recomendaciones efectuadas, el análisis continuará por ese camino.

4. Diseño del sistema recomendado

En esta cuarta fase, el analista se basa en la información recopilada previamente con el fin de hacer el diseño lógico del SI. La misión del analista en esta etapa es diseñar los ordenamientos para ayudar a que los usuarios introduzcan los datos correctamente. Del mismo modo, el analista debe colaborar para que los usuarios realicen la entrada de datos mediante técnicas del buen diseño de formularios y páginas Web o pantallas.

Esta fase también considera el diseño de bases de datos, las que contendrán muchos de los datos requeridos por quienes toman decisiones en la empresa, además de diseñar los controles y procedimientos de respaldo para resguardar el sistema y los datos.

5. Desarrollo y documentación del software

En esta quinta fase, el analista deberá trabajar en conjunto con los programadores, con el fin de desarrollar el software original solicitado por los usuarios. Del mismo modo, la misión común del analista y los usuarios será desarrollar documentación efectiva para el software (manuales, ayudas en línea, FAQs o Léeme). Esta fase debe considerar las preguntas realizadas por los usuarios y trabajadas junto con el analista. No está de más señalar que la documentación del software orientará a los usuarios respecto al uso del mismo y cómo solucionar eventuales problemas.

En esta fase, los programadores serán los encargados de analizar los errores sintácticos que existan en los programas, con el fin de asegurar la calidad del producto. Así, el programador deberá “zambullirse” en la lectura del código y el diseño con el fin de explicar de la mejor manera los fragmentos complejos del programa a un equipo de otros programadores.



6. Prueba y mantenimiento del sistema

En esta sexta fase, los programadores deberán probar el sistema de información, con el fin de detectar problemas antes de entregar el producto. Si bien los programadores realizan solos esta tarea, luego se incluirá a los analistas de sistemas.

Los planes de prueba suelen ser creados durante las primeras etapas del ciclo de vida. Estos van modificándose y mejorándose mientras el proyecto avanza. Es importante señalar que el mantenimiento del sistema (junto con toda la documentación del mismo) se inicia en esta fase; y, además, se realizará constantemente durante toda la vida del SI. Así, la mayor parte del trabajo que deberá realizar el programador consiste en labores de mantenimiento. Una buena parte de los procedimientos sistemáticos realizados por el analista durante el ciclo de vida de un SI colaboran para asegurar que este mantenimiento se mantenga en un nivel mínimo necesario.

7. Implementación y evaluación del sistema

En esta séptima y última fase, el analista tendrá la labor de colaborar con la implementación del sistema de información. Por tanto, en esta etapa es necesario capacitar a los usuarios con el fin de que puedan operar el sistema sin problemas. Los distribuidores se encargarán de una parte de la capacitación; no obstante, el analista de sistemas será el encargado de supervisar la capacitación.

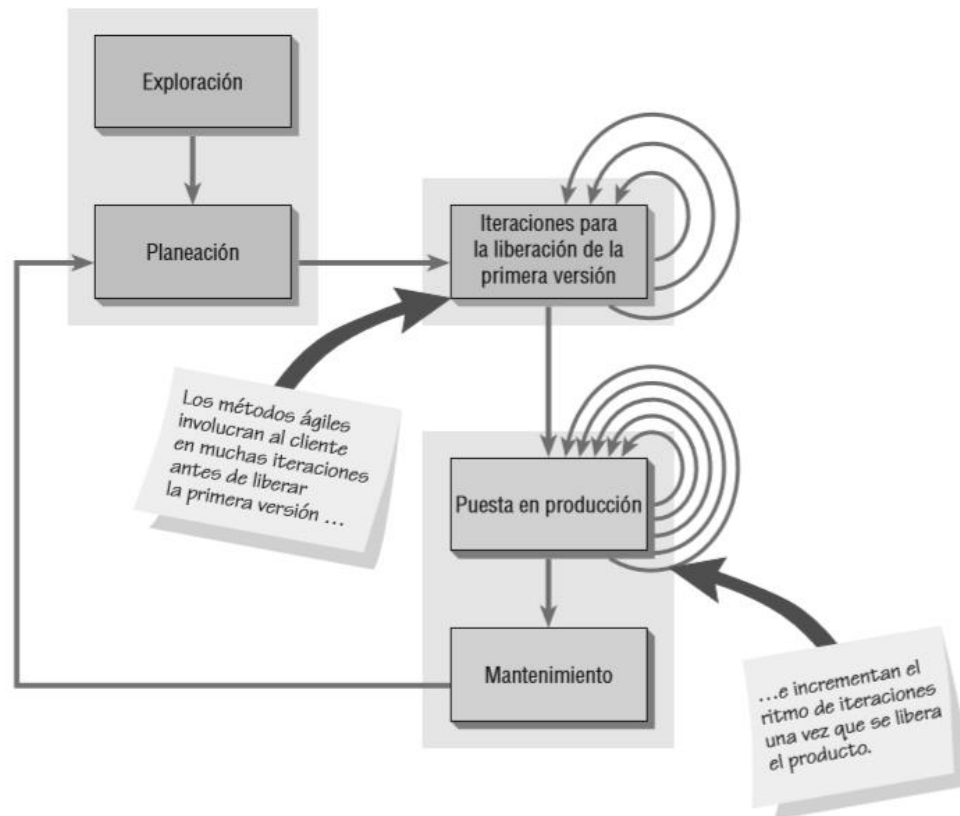
Del mismo modo, es muy importante que el analista planifique el cambio desde el sistema antiguo al nuevo, con el fin de que no existan imprevistos. Este proceso requiere actualizar los formatos de los archivos o crear una base de datos, junto con instalar equipos y llevar el nuevo sistema a producción.

Modelos para planificar el proceso

Existen distintos modelos para planificar el proceso de desarrollo de software. Cada uno de estos modelos tienen características que los hacen más adaptables a ciertos tipos de proyectos.

Es posible clasificar estos modelos en dos: metodologías secuenciales y metodologías ágiles.

- **Metodologías secuenciales:** también conocidas como "tradicionales". Consisten en varias etapas que se van desarrollando de manera ordenada una tras otra. Este tipo de metodologías fueron las primeras empleadas en proyectos de desarrollo de software y hasta hoy en día son ampliamente utilizadas.
- **Metodologías ágiles:** se desarrollan de manera iterativa, repitiéndose etapas cortas en las que se realizan pequeñas secciones del proyecto. Las metodologías ágiles se caracterizan por dos elementos: ser **interactivas** e **incrementales**. Posee cinco etapas: exploración, planeación, iteraciones para la liberación de la primera versión, puesta en producción y mantenimiento. Revisémoslas a continuación:



Las cinco etapas del proceso de desarrollo de modelado ágil muestran que las iteraciones frecuentes son esenciales para un desarrollo exitoso del sistema. Fuente: Kendall y Kendall, 2011.

Las tres flechas circulares que van de regreso hacia la caja de “iteraciones” representan aquellos **cambios incrementales** provenientes de procesos repetidos de prueba y retroalimentación, conducentes hacia un sistema estable y evolutivo. Por cierto, el número de iteraciones subirá tras la liberación del producto. Tras el mantenimiento se vuelve a planeación, por lo que existe un ciclo de retroalimentación constante compuesto por usuarios (clientes) y el equipo de desarrollo.

Estas etapas nos demuestran que las iteraciones frecuentes son importantísimas para el correcto y eficiente desarrollo del sistema.



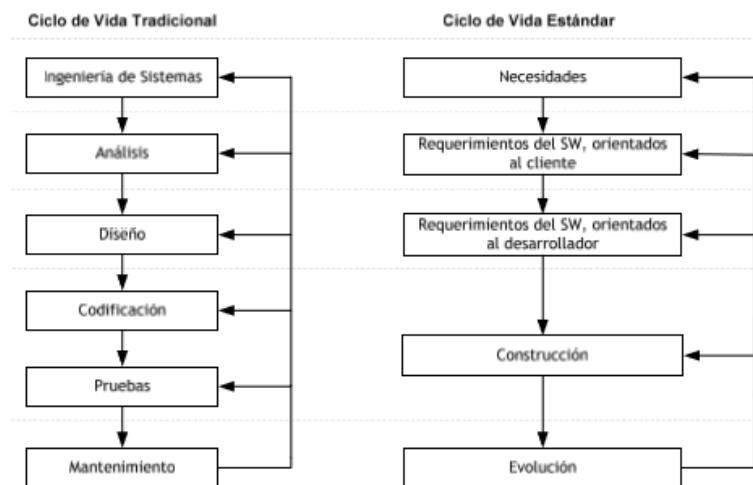
Como puedes ver, una diferencia fundamental entre ambas metodologías consiste en que en la metodología secuencial el producto será utilizable solo al momento en que concluyan todas las etapas del proyecto; mientras que en la metodología ágil se trabaja de forma iterativa, situando cada parte y haciendo el proyecto cada vez más funcional.

Algunos modelos secuenciales

a) Ciclo de vida tradicional o modelo en cascada (Royce, 1970)

Está basado en la producción secuencial de tipos de productos en distintos niveles de abstracción. Este modelo implica transformar de forma lineal todos los requerimientos iniciales hacia un producto que se va haciendo más concreto.

Además de ser el modelo más antiguo y usado, es preciso destacar que no sigue un flujo necesariamente secuencial (existen iteraciones). Así, se torna complejo para el usuario establecer cada uno de los requerimientos de forma explícita. Si bien es un modelo antiguo, es útil para comenzar.



Ciclo de vida tradicional v/s ciclo de vida estándar. Fuente: Bedini, 2005.

Ejemplo de proyecto que utiliza esta metodología por sus características

Las metodologías de desarrollo tradicionales se emplean, sobre todo, en proyectos de arquitectura e ingeniería. Estas especialidades requieren un diseño muy fijo y exhaustivo antes del comienzo de fases posteriores. Por ejemplo, un edificio que se construye no puede iterar respecto a sus cimientos o el diseño estructural del primer piso. Podemos percatarnos, entonces, que el desarrollo de software presenta **características muy distintas** a aquellos otros tipos de proyectos.

Estas metodologías tradicionales intentan predecir las variables del proyecto. Asimismo, los clientes no podrán ver el producto hasta que esté terminado o construido.

Finalmente, es importante destacar que si existen retrasos en alguna etapa del desarrollo, se provoca un retraso en el proyecto completo.

```
graph LR; MR[MODELADO DE LOS REQUERIMIENTOS] --> DA[DISEÑO DE LA ARQUITECTURA]; DA --> DC[DISEÑO DE LOS COMPONENTES]; DC --> GC[GENERACIÓN DE CÓDIGO]; GC --> PU[PRUEBAS UNITARIAS]; MR --> PA[PRUEBAS DE ACEPTACIÓN]; MR --> PS[PRUEBAS DEL SISTEMA]; DA --> PS; DA --> PC[PRUEBAS DE COMPONENTES]; DC --> PC; DC --> PU; GC --> PU; PA --> MR; PS --> MR; PS --> DA; PC --> DA; PC --> DC; PU --> DC; PU --> GC;
```

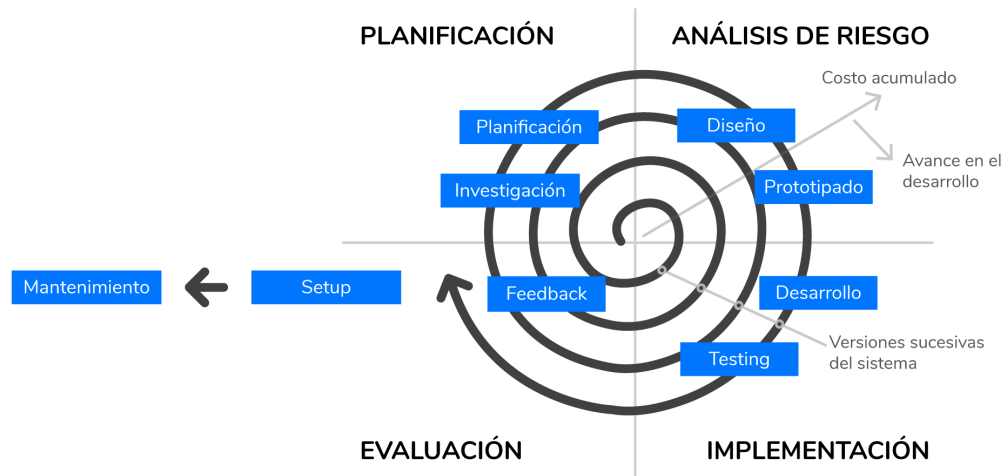
Mientras descendemos por el lado izquierdo de la V, los requerimientos básicos se dirigen hacia representaciones técnicas cada vez más específicas. Tras la generación del código, debemos subir por el lado derecho de la V para realizar pruebas que validen los elementos creados durante el descenso por el lado izquierdo de la V.

El modelo V funciona en diverso tipo de contextos, como al atender problemas complejos que sabemos que sucederán. Es el caso, por ejemplo, de los procesos educativos en instituciones de educación superior, muy complejos por la gran variedad de elementos que los componen (organizados en distintos niveles, que interactúan entre sí y con otros factores externos, etc.). Así, este método le otorga importancia a la calidad y superación del producto, el que se va evaluando en distintas etapas de su desarrollo.

Algunos modelos evolutivos

a) Modelo espiral (Boehm, 1986)

Este modelo está orientado a determinar riesgos en el desarrollo del software. En este caso, el desarrollo iterativo se planifica en forma de espiral.



Fuente: aspgems.com, s.f.

Como es posible apreciar en la figura, los ciclos interiores simbolizan el análisis inicial del proyecto, mientras que el prototipado y el resto de ciclos exteriores simbolizan un ciclo de vida clásico. Este modelo es apropiado para grandes proyectos, ya que se combina con análisis de riesgo en cada ciclo.

Sus actividades principales son:

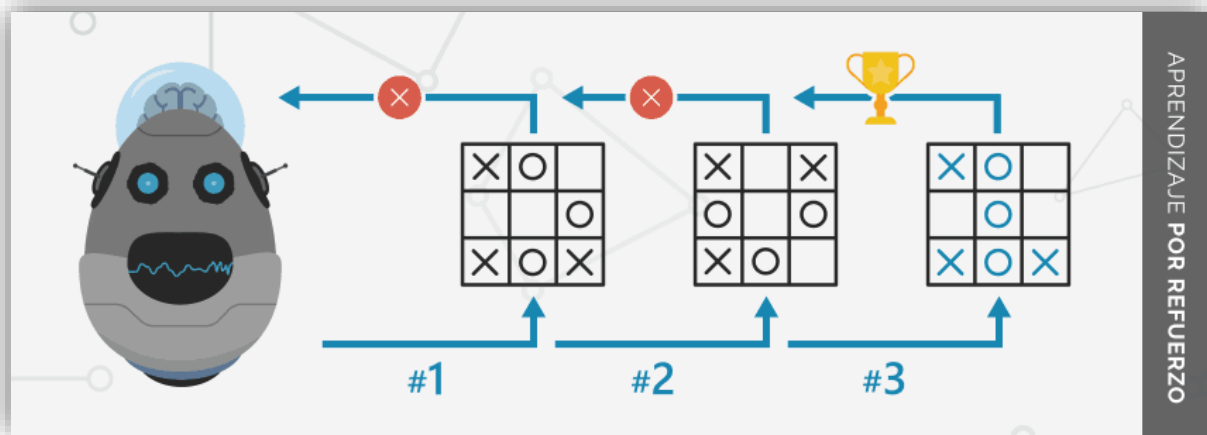
- **Planificación:** objetivos, alternativas, restricciones.
- **Análisis de riesgo:** estudio de alternativas y riesgos.
- **Implementación:** desarrollo del producto.
- **Evaluación:** realizada por el cliente.

Ejemplo de proyecto que utiliza esta metodología por sus características

Reinforcement Learning, o aprendizaje por refuerzo.

El **aprendizaje por refuerzo** es lo que conocemos como **ensayo y error**. En términos prácticos, un software, máquina o robot será capaz de aprender y ejecutar, por ejemplo, las reglas de un juego en un tiempo muy corto con el fin de alcanzar un determinado objetivo.

Este verdadero **aprendizaje** conduce al programa a obtener resultados con base en la experiencia previa. Existe un gran número de programas y máquinas que utilizan esta metodología, sobre todo en el ámbito de la inteligencia artificial y el ajedrez (con AlphaZero y DeepMind como sus mayores referentes).



Fuente: auraportal.com, s.f.

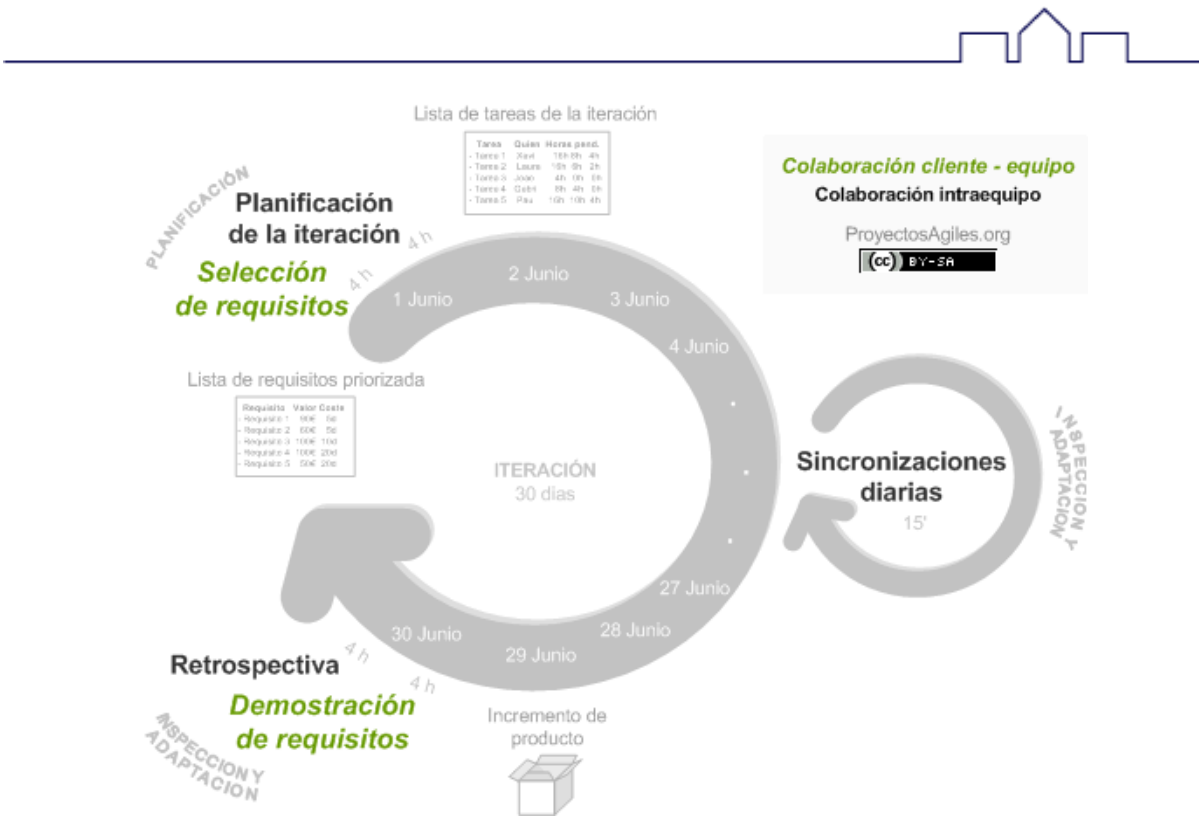
b) Metodología Scrum (Nonaka y Takeuchi, principios de los 80)

Scrum es un modelo para trabajo colaborativo que se basa en iteraciones. Al ser una metodología ágil, se utiliza para desarrollar proyectos con un gran número de cambios imprevistos o de última hora. Su objetivo es guiar actividades de desarrollo en procesos que poseen determinadas actividades en su estructura: requerimientos, análisis, diseño, evolución y entrega.

En Scrum, cada proyecto se realiza en ciclos de tiempo reducidos y de duración fija, con iteraciones que duran aproximadamente dos semanas, sin embargo, en algunos equipos consideran tres e incluso cuatro semanas, periodo máximo de retroalimentación del producto real y una posterior reflexión. Cada iteración debe entregar un resultado completo., Se va revisando el trabajo validado de la semana anterior debido a que los requerimientos van cambiando a corto plazo, sobre esto se priorizan y planifican las actividades en las que se invertirán los recursos en la próxima iteración.

En cada actividad estructural, las tareas del trabajo ocurren con un patrón del proceso llamado sprint². El trabajo realizado dentro de ellos se adapta al problema en cuestión y se define en tiempo real por parte del equipo Scrum, tiempo que generalmente se modifica.

² Sprint: nombre que recibe cada ciclo o iteración dentro de un proyecto Scrum. La cantidad de ellos que se requiere para cada actividad estructural es diferente de acuerdo a la complejidad y el tamaño del proyecto.



Fuente: proyectosagiles.com, s.f.

Ejemplo de proyecto que utiliza esta metodología por sus características

Proyectos encargados del desarrollo de videojuegos (VJ).

En el caso del desarrollo de videojuegos, la utilización de Scrum resulta de gran utilidad durante el desarrollo del producto. Esto sucede debido a ciertas características que posee dicha metodología, como la flexibilidad ante los cambios y la importancia que posee la calidad del software (incluso tras la finalización y comercialización del producto).

Hoy en día, los **parches** posteriores al lanzamiento de un producto son algo común y natural. Por lo mismo, el empleo de una metodología ágil como Scrum es de vital importancia a la hora de generar estos cambios.

Asimismo, Scrum permite reducir riesgos y predecir con mayor exactitud los tiempos de desarrollo del producto. Al trabajar los equipos por *sprint*, es muy factible estimar el tiempo de trabajo durante cada *sprint*.

A modo de conclusión, revisemos una tabla con los ciclos de vida más importantes:

Predictivos	Iterativos	Incrementales	Ágiles
Los requisitos son definidos por adelantado antes de que comience el desarrollo	Los requisitos pueden ser elaborados a intervalos periódicos durante la entrega	Los requisitos se elaboran con frecuencia durante la entrega	
Entregar planes para el eventual entregable. Posteriormente, entregar solo un único producto final al final de la línea de tiempo del proyecto	La entrega puede ser dividida en subconjuntos del producto global	La entrega ocurre frecuentemente con subconjuntos del producto global valorados por el cliente	
El cambio es restringido tanto como sea posible	El cambio es incorporado a intervalos periódicos	El cambio es incorporado en tiempo real durante la entrega	
Los interesados clave son involucrados en hitos específicos	Los interesados clave son involucrados periódicamente	Los interesados clave son involucrados continuamente	
El riesgo y los costos son controlados mediante una planificación detallada de las consideraciones que mayormente se conocen	El riesgo y los costos son controlados mediante la elaboración progresiva de los planes con nueva información	El riesgo y los costos son controlados a medida que surgen los requisitos y limitaciones	

Características de los ciclos de vida. Fuente: PMBOK, 2017.

Calidad dentro del desarrollo de un sistema de información

El ciclo de vida básico clásico de un sistema consta de siete etapas. En cada una de ellas es posible considerar elementos que van a permitir evaluar la calidad del software. Dentro de ellos se encuentran:

1. Precisar los elementos de un sistema de gestión de calidad.
2. Documentar el sistema de calidad.
3. Soporte del control y garantía de la calidad.
4. Determinar formas de revisión para el sistema de gestión de calidad.
5. Determinar formas de control.
6. Establecer procedimientos para la corrección.

Garantía de la calidad del software

La garantía de la calidad corresponde a un grupo de tareas de auditoría e información que evalúan la efectividad del sistema, es decir, el nivel de certeza de que el sistema no presenta vulnerabilidades (ya sea que hayan sido diseñadas de manera intencional o se hayan incorporado accidentalmente en cualquier etapa de su ciclo de vida). Del mismo modo, se evalúa que el sistema funcione como se tiene previsto. En el aseguramiento de la calidad se le presentan al gestor los datos necesarios para que pueda evaluarlos. Es responsabilidad de él estudiarlos y utilizar los recursos necesarios para resolver los conflictos y lograr la calidad.

Calidad de software

La calidad del software se puede observar en una característica o atributo. Se refiere a características que se pueden medir, es decir información que se puede comparar al conocer patrones definidos, como tamaño, color y elasticidad.

En las características medibles se pueden encontrar dos tipos de calidad:

- a) **Calidad de diseño:** corresponde a la realización de un diseño ajustado de forma correcta a las necesidades del cliente.
- b) **Calidad de concordancia:** corresponde al grado de cumplimiento de las especificaciones y aspectos indicados durante la "calidad de diseño". El grado de cumplimiento es directamente proporcional a la calidad de concordancia, es decir, entre mayor sea el grado de cumplimiento, la calidad de concordancia será mucho mayor porque se estará cumpliendo con lo especificado.

El control de calidad incluye inspecciones, revisiones y pruebas utilizadas durante todo el proceso de desarrollo de un sistema, por medio del cual se garantiza que el producto final cumpla con los requisitos que se han establecido inicialmente.

Costo de la calidad

Incluye todos los costos que se generan o que se requieren en el desarrollo de las actividades relacionadas con la calidad. Los costos de la calidad se dividen en:

- a) Costos asociados con prevención.
- b) Evaluación y fallas.

¡REVISEMOS LO APRENDIDO!



BASE DE DATOS

Colección de información organizada de forma que los programas que están en los computadores puedan tomar rápidamente aquellos fragmentos de datos que necesitan para su funcionamiento.



MODELADO DE DATOS

Forma de establecer una estructura y organización de los datos para que puedan ser accedidos fácilmente por las bases de datos.



DATO

Representación simbólica (numérica, alfabética, algorítmica, etc.) de un atributo de una entidad.



ENTIDAD

Objeto que existe en el mundo real y que se diferencia de otros objetos.



MODELO DE NEGOCIOS

Un modelo de negocio es una herramienta anterior al plan de negocio que define claramente qué se va a ofrecer al mercado, cómo se hará, quién será el público objetivo, cómo se va a vender y cuál será la forma para generar ingresos.

Los elementos básicos que debe contener un modelo de negocio son:

- Cuáles son y cómo se va a relacionar con los clientes.
- Cuál es la propuesta de valor³.
- Cómo va a desarrollar el producto o servicio.
- Cuánto va a costar esto (inversión inicial).

Un modelo de negocio se caracteriza por abarcar varios factores, tales como definir las particularidades de los productos a vender, definir cómo llegar a nuestros clientes, esbozar cómo promocionar nuestro producto y el objetivo del producto, entre otros.

En otras palabras, realizar un modelo de negocio significa plasmar en un documento cómo se va a crear, desarrollar y capturar valor. Asimismo, simboliza una visión de lo que puede llegar a ser el negocio y en qué bases se sostendrá una empresa (o los pilares del edificio que queremos construir junto al cliente).

Modelo de negocio Canvas

Canvas es un tipo de modelo que cubre todos los aspectos de un negocio. Se caracteriza por abarcar desde clientes hasta socios clave, pasando por costos y otros elementos. Su objetivo es condensar en un solo sitio la forma en que se crea, entrega y captura valor de la puesta en marcha (*start up*).

³ Factor que influye para que un cliente se decida por una u otra empresa. El propósito de una propuesta de valor es solucionar un problema o satisfacer una necesidad de nuestro cliente.



Segmentos de clientes

Consiste en segmentar el mercado o grupo de personas a quienes se venderá el producto o servicio. Es posible clasificar a los clientes de acuerdo a sus necesidades, canales, relaciones u ofertas. Ejemplos de segmentos de clientes: mercado de masas (amplio), nichos de mercado (estrecho), diversificados (público heterogéneo) o multi-segmentos (dependen de muchos clientes al mismo tiempo).

PREGUNTAS ORIENTADORAS	PERMITEN IDENTIFICAR
<ul style="list-style-type: none">- ¿Para quién estamos creando valor?- ¿Quiénes son nuestros clientes más importantes?	<ul style="list-style-type: none">- Necesidades de cada segmento de clientes.- Diferentes canales de distribución.- Diferentes tipos de relación.- Diferentes márgenes de beneficios.

Fuente: elaboración propia

Propuesta de valor

Son las características y beneficios encargadas de crear valor para cada segmento. En esta fase es preciso explicar lo ofrecido y por qué los clientes deberían comprarlo. Ejemplos: novedad, rendimiento, personalización, diseño, precio, entre otros.

PREGUNTAS ORIENTADORAS	PERMITEN IDENTIFICAR
<ul style="list-style-type: none">- ¿Qué valor entregamos al cliente?- ¿Cuál de los problemas de nuestro cliente solucionamos?- ¿Qué necesidad estamos satisfaciendo?	Novedad, diseño, precio, marca, mejoras en productos o servicios y una propuesta de valor diferente para cada segmento.

Fuente: elaboración propia



Canales

Medios a través de los cuales se comunica la propuesta de valor al cliente. Los canales pueden ser propios o externos. Se dividen en: notoriedad, evaluación, compra, entrega y postventa.

PREGUNTAS ORIENTADORAS	PERMITEN IDENTIFICAR
<ul style="list-style-type: none">– ¿Por qué canales prefieren mis clientes ser contactados?– ¿Cómo estamos contactando con ellos ahora?– ¿Cuál es el canal que mejor funciona?– ¿Cuál es el canal más eficiente en coste?– ¿Cómo lo integramos en la rutina de los clientes?	<ul style="list-style-type: none">– Comunicación, distribución y canales de ventas.– Ayudar al cliente a que tome conciencia de nuestra propuesta de valor.– Ayudar al cliente a evaluar la propuesta de valor.– Facilitar la compra al cliente.– Servicio post venta.

Fuente: elaboración propia

Relación con el cliente

Tipo de relación entre la empresa emergente (*start up*) y el cliente. Ejemplos: asistencia personal, *self-service* o automatizado (mezcla de ambas).

PREGUNTAS ORIENTADORAS	PERMITEN IDENTIFICAR
<ul style="list-style-type: none">– ¿Qué tipos de relación queremos establecer y mantener con nuestro cliente?– ¿Cuánto cuesta esta relación?– ¿Cómo se integra en nuestro modelo negocios?	<ul style="list-style-type: none">– Relación personal o relaciones humanas.– Relación personal con dedicación.– Self-service (cliente se sirve solo).– Servicios automatizados, relación con grabaciones, maquinas, S.O.– Communities (clientes se relacionan entre ellos).– Concreciones.

Fuente: elaboración propia



Fuente de ingresos

¿Cómo llega el dinero y se generan beneficios? Ejemplos: venta directa en un único pago, pago por uso o suscripción, entre otros.

PREGUNTAS ORIENTADORAS	PERMITEN IDENTIFICAR
<ul style="list-style-type: none">– ¿Por cuál propuesta de valor están dispuestos a pagar nuestros clientes?– ¿Por qué están pagando actualmente?– ¿Cómo están pagando actualmente?– ¿Cómo preferirían pagar?– ¿Cuál es el porcentaje de ingresos de cada línea de ingreso respecto a los ingresos totales?	<ul style="list-style-type: none">– Ingresos por cada segmento de cliente.– Pagos de una vez.– Pagos recurrentes, por servicio.– Pagos de servicios post venta.

Fuente: elaboración propia

Recursos clave

Corresponden a los recursos imprescindibles para que funcione todo lo mencionado anteriormente. Pueden ser físicos, intelectuales, humanos o financieros. Ejemplos: vehículos, patentes, ejecutivos importantes o el ingreso de dinero.

PREGUNTAS ORIENTADORAS	PERMITEN IDENTIFICAR
<p>¿Qué recursos clave requiere nuestra propuesta de valor, nuestros canales de distribución, la relación con el cliente y las fuentes de ingreso?</p>	<ul style="list-style-type: none">– Recursos propios, alquilados o comprados a otros.– Recursos físicos, intelectuales, humanos, financieros.– Marcas, patentes, etc.

Fuente: elaboración propia



Actividades clave

Son las actividades imprescindibles para que el negocio prospere. Se deberá definir si corresponden a de producción, de solución a problemas o de una plataforma a través de la que funciona la empresa emergente (*start up*).

Fuente: elaboración propia

PREGUNTAS ORIENTADORAS	PERMITEN
<ul style="list-style-type: none">– ¿Qué actividades clave intervienen en nuestra propuesta de valor?– ¿Qué actividades clave requieren nuestros canales de distribución?– ¿Qué actividades clave requiere nuestra relación con los clientes?– ¿Qué actividades clave requiere nuestra fuente de ingresos?	<ul style="list-style-type: none">– Producir.– Atender clientes.– Solucionar problemas.– Plataforma web, networking.

Socios clave

Son aquellos colaboradores clave para que todo marche como se debe. Los socios clave son necesarios para optimizar recursos, reducir riesgos con alianzas estratégicas o adquirir recursos y actividades que no existen en la propia empresa emergente (*start up*).

PREGUNTAS ORIENTADORAS	PERMITEN
<ul style="list-style-type: none">– ¿Quiénes son nuestros socios?– ¿Quiénes son nuestros proveedores?– ¿Cuáles son los recursos que estamos adquiriendo de los socios?– ¿Cuáles son las actividades que hacen los socios?	<ul style="list-style-type: none">– Descripción del network de proveedores y socios, que hacen que el modelo de negocios funcione y se optimice, se reduzcan riesgos y se adquieran recursos clave.– Optimizar economías de escala.– Reducir riesgos e incertidumbres.– Adquirir un particular activo o recurso.

Fuente: elaboración propia



Estructura de costos

Corresponde al desglose de los gastos que el modelo de negocio sostendrá. Ejemplos: costos fijos, variables, reducción de costos y todo lo relacionado con gastos.

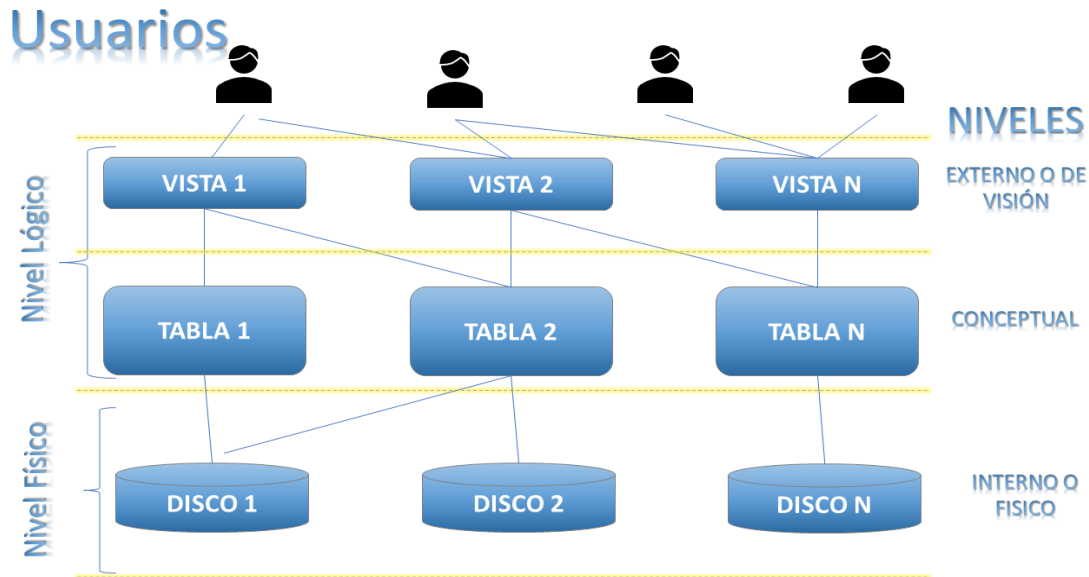
PREGUNTAS ORIENTADORAS	PERMITEN
<ul style="list-style-type: none">– ¿Cuáles son los costos más importantes de nuestro modelo de negocio?– ¿Cuáles son los recursos más caros?– ¿Cuáles son las actividades más caras?	<ul style="list-style-type: none">– La descripción de todos los costos en los que incurre la empresa para desarrollar su modelo (cost driven, automatizado, simplificado).– Value driven, valor para el cliente.– Costos fijos y variables.– Economías de escalas.

Fuente: elaboración propia

Las descripciones de los conceptos de modelo de negocio permitirán comprender las reglas en un sistema de información. Evitará al profesional que efectúe el proceso de toma de requerimientos omitir etapas del proceso que tenga una organización en su estructura comercial. Le permitirá, por su parte, efectuar un análisis correcto y real de los actores involucrados, y describir los alcances potenciales del sistema de información desarrollar.

ARQUITECTURA DE UN SISTEMA DE BASE DE DATOS

Mencionamos anteriormente que una arquitectura de tres niveles se enfoca en separar los programas de aplicación de la base de datos física.



Fuente: elaboración propia.

Por medio de la imagen se pueden apreciar los tres niveles: físico que incluye los discos (donde se almacena la información), conceptual (las tablas dentro de las bases de datos, la estructura en sí) y externo (vistas, que corresponde a la forma en que son presentados los datos al usuario, por ejemplo, las pantallas), sobre las vistas en la imagen se encuentran los usuarios, que son quienes interactúan directamente con el sistema.

Objetivos de los DBMS

Database Management System, en inglés; o **Sistema Gestor de Base de Datos**, en español. Corresponden a programas nos permitirán almacenar, modificar y extraer información en una base de datos. Los usuarios pueden acceder a la información utilizando herramientas concretas de consulta y de creación de informes, o bien por medio de aplicaciones.



Los objetivos de los SGBD son:

- **Evitar redundancia de datos:** refiere a almacenar datos similares muchas veces en distintos lugares.
- **Evitar inconsistencia de datos:** sucede cuando las copias redundantes contienen distinta información.
- **Permitir en todo momento el acceso a los datos.**
- **Ante el acceso concurrente⁴, evitar anomalías.**
- **Seguridad y restricción ante accesos no autorizados.**
- **Integridad de datos:** consiste en que la información en una base de datos sea correcta y esté completa.
- **Backup (respaldos):** refiere a la copia de seguridad realizada para recuperar los datos originales en caso de pérdida.

Independencia lógica y física de los datos

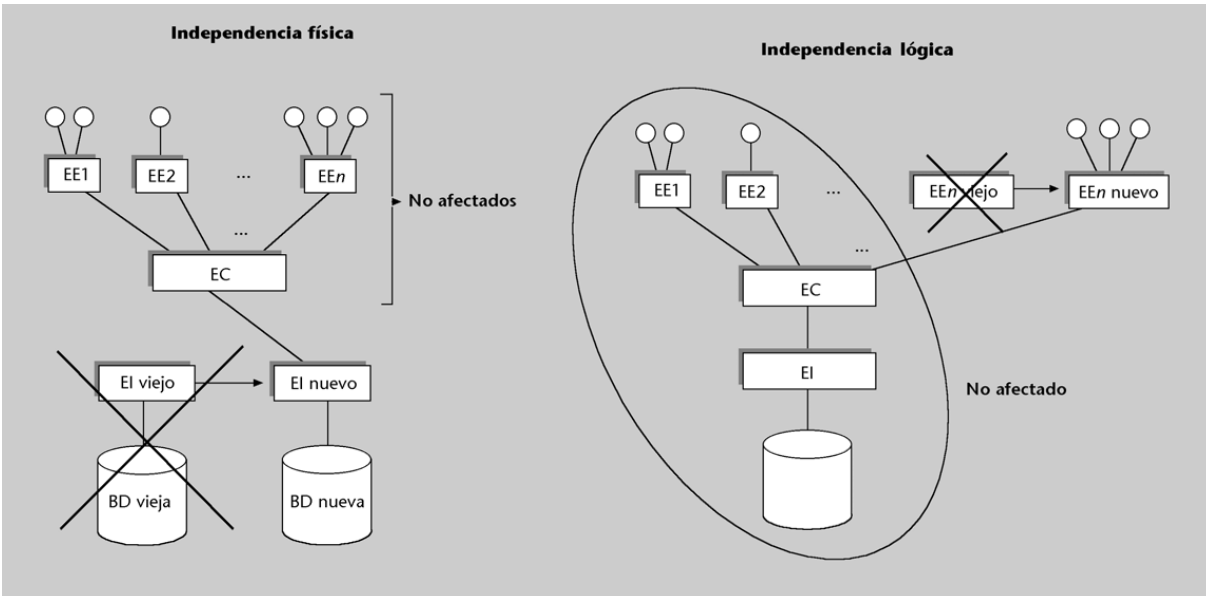
Para continuar, estudiaremos la **independencia lógica y física de los datos**. El primer concepto refiere a si se realiza alguna modificación de la estructura de la base de datos (por ejemplo, ampliar o reducir la base de datos), las vistas que poseen las aplicaciones (esquemas externos o programas de aplicación) no tendrían que sufrir alteraciones ni verse afectadas. A eso denominaremos **independencia lógica**. Si, en cambio, el esquema lógico permanece invariable al reemplazar un disco duro o el sistema operativo, estamos observando únicamente la separación entre las aplicaciones y las estructuras físicas de almacenamiento. A esto denominaremos **independencia física**.

En otras palabras, la **independencia física** corresponde a la **capacidad de modificar el esquema físico** de los datos sin que deban volverse a realizar todos los programas de la aplicación (es decir, no debe verse afectado el esquema conceptual).

La **independencia lógica** de datos, por su parte, refiere a modificar el esquema conceptual pero provocando **que no se reescriban los programas de aplicación**. Un ejemplo de esto es la necesidad de reordenar algunos ficheros físicos para mejorar el rendimiento.

Se entiende que las independencias lógicas son mucho más complejas de lograr que las independencias físicas, debido a que los programas de aplicación suelen ser mucho más dependientes de la estructura lógica de los datos a los que acceden.

⁴ Acceso concurrente: Un sistema que permita a varias estaciones de trabajo modificar en forma simultánea una misma base de datos, debe tomar precauciones para evitar operaciones concurrentes sobre un mismo registro. Bloquea el registro usado en ese momento como medida de seguridad.



Fuente: dataprix.com, s.f.

Ahora que hemos revisado algunos conceptos generales, veremos todo lo relacionado con la captura requerimientos de clientes reales.

Requerimientos de clientes

Todos los modelos del proceso de desarrollo de software contienen ciertas actividades que están destinadas a la toma de requerimientos. Así, para entender la finalidad y eventual operación del sistema debemos realizar un **análisis de los requerimientos**.

Definición de requerimiento

Es muy frecuente que el cliente que solicite un programa tenga ciertas nociones de lo que este debe hacer. A esto podemos definirlo como el **propósito** del programa.

Los requerimientos son declaraciones que identifican una característica, un atributo o descripción que puede representar una capacidad o condición que debe cumplir un sistema (o un software) para que el usuario lo considera valioso. El objetivo es satisfacer el propósito del sistema. Un requerimiento, entonces, se relaciona con lo que el cliente desea que realice el sistema.

Por tal motivo, el analista debe ser capaz de comprender el problema de los usuarios teniendo en cuenta su cultura y lenguaje, con el fin de, posteriormente, realizar el sistema que resuelva los problemas o necesidades del cliente.



Requerimiento v/s diseño

- El requerimiento define el problema.
- El diseño define la solución.

Es importante destacar que el análisis de requerimientos considera solamente el problema del cliente, y no los elementos más específicos de la implementación (salvo que el cliente así lo solicite). Por ejemplo, podemos considerar elementos específicos una determinada base de datos o cierto lenguaje de programación que el cliente solicite emplear.

Importancia de los requerimientos

Existe bastante experiencia acumulada como para afirmar que una falta de estudios previos o la carencia de una definición del alcance del proyecto provocan que muchos proyectos fracasen incluso antes de empezar.

Por ejemplo, el modelado del negocio es fundamental; si no se realiza, implica que el analista no está enterado o involucrado en el problema. Esto es muy importante, ya que podemos tener claridad respecto al tipo de sistema que queremos desarrollar, pero si no conocemos a cabalidad el problema, corremos el riesgo de que los requisitos que hemos identificado no estén destinados a resolver las necesidades del cliente.

Del mismo modo, si no involucramos al cliente de forma activa, podemos tener serios inconvenientes. Una solución es modelar junto al usuario.

Finalmente, si existen requerimientos incompletos o que cambian con demasiada frecuencia, lo más probable es que nuestro proyecto fracase. Por lo tanto, resulta fundamental que los requerimientos estén bien definidos desde un inicio. Así, podremos evitar problemas como los que se presentan a continuación:

- ✗ Se hacen estimaciones que no son realistas.
- ✗ Se utilizan de manera incorrecta herramientas para planificar.
- ✗ No es posible hacer revisiones frecuentes del progreso.
- ✗ Las bases de la arquitectura, el diseño y el desarrollo son poco sólidas.
- ✗ Las pruebas se realizan sobre suposiciones y no respecto a los requerimientos del cliente.
- ✗ Los requerimientos aumentan y se tornan incontrolables.

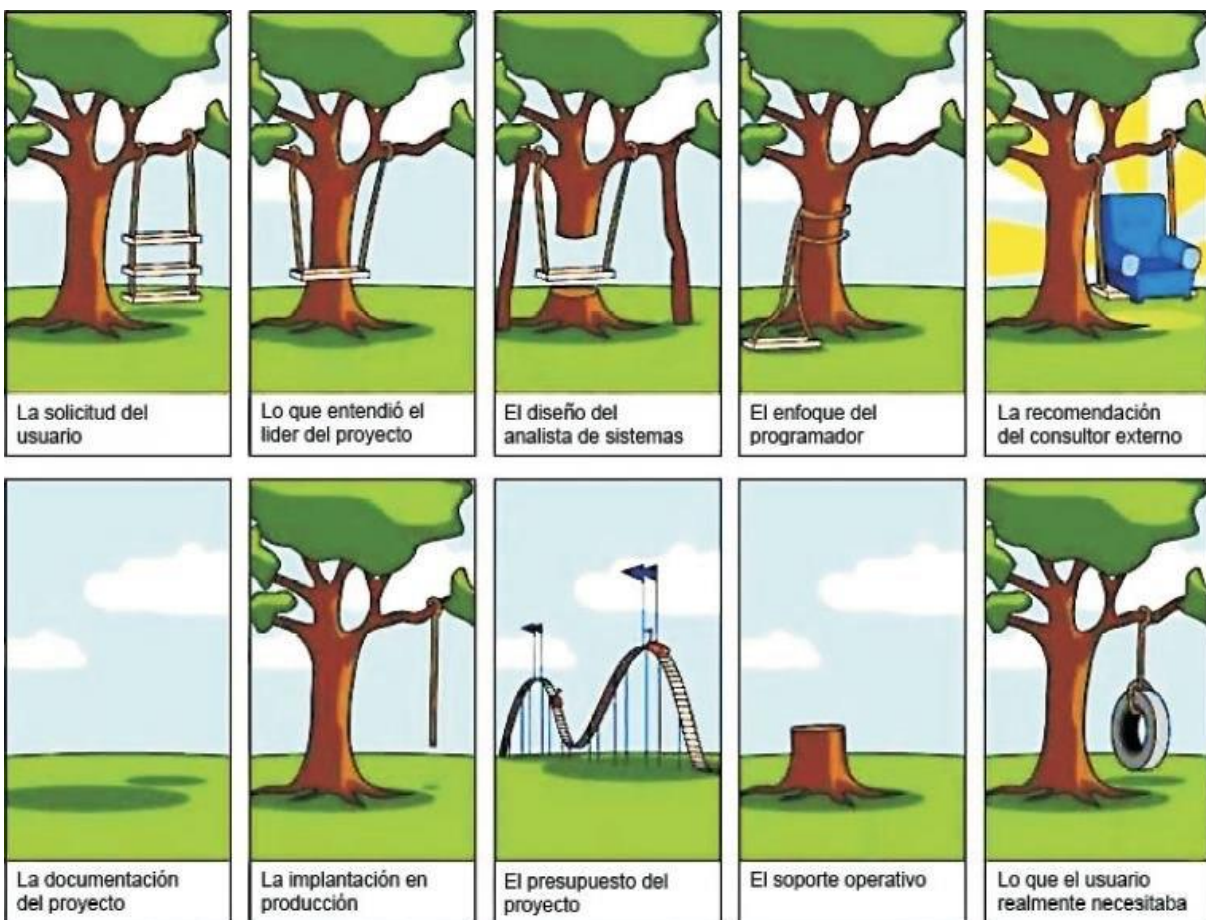


Entonces, es posible concluir que la importancia de los requerimientos reside en que simbolizan el hilo conductor del desarrollo del software.

Los requerimientos de calidad son, entonces, importantísimos. El éxito de nuestro proyecto se deberá a múltiples factores, entre los que destacan:

1. El uso adecuado de técnicas de captura de requerimientos.
2. La experiencia de trabajo del analista.

De este segundo punto se desprende que nuestra experiencia, por ejemplo, tendrá importancia al momento de elegir una determinada técnica al momento de entrevistar a nuestro cliente. Si aquel cliente no comprende nada respecto al lenguaje informático, debemos ser capaces de emplear un lenguaje claro y suficientemente preciso en el momento de la entrevista.



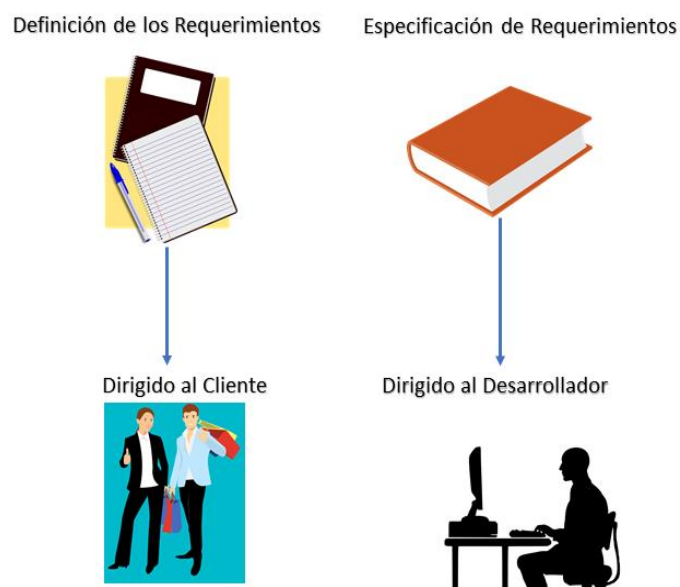
Fuente: itsoftware.com.co, s.f.

Documentos de requerimientos

Para el análisis de requerimientos es necesario contar con estos dos documentos:

- **Documento de definición de requerimientos:** lista de aquello que el cliente necesita que el sistema realice. Debe ser escrito para que el cliente lo comprenda claramente. Este documento es redactado colaborativamente entre el cliente y el analista.
- **Documento de especificación de requerimientos:** traduce lo redactado en el documento anterior y lo hace legible en un lenguaje técnico para el desarrollador que diseñará el sistema. Este documento será redactado exclusivamente por el analista de requerimientos.

Si bien puede ocurrir que un solo documento sirva para todas las partes (cliente, analista y desarrollador), lo más común es que se requieran los dos documentos antes mencionados.



Fuente: elaboración propia.

Resulta significativo que, al utilizar estos documentos, exista una relación entre cada requerimiento del documento de definición y los documentos en la especificación. De esta manera, se consigue unir el enfoque del cliente y el de los desarrolladores.

Finalmente, es importante mencionar que la definición de los requerimientos suele estar redactada en nuestro lenguaje convencional (el español, en este caso). No obstante, la especificación de dichos requerimientos se escribe de manera mucho más técnica. Así, un requerimiento realizado en lenguaje convencional puede ser traducido, por ejemplo, a un diagrama de flujo de datos o a una ecuación.



Clasificación de requerimientos

Los requerimientos se clasifican según su objetivo o a quién van dirigidos:

Según objetivo	Según a quién van dirigidos
<ul style="list-style-type: none">• Requerimientos funcionales.• Requerimientos no funcionales.• Requerimientos del dominio.	<ul style="list-style-type: none">• Requerimientos del usuario.• Requerimientos del sistema.

A continuación, revisaremos en detalle esta clasificación.

Según tipo de requerimiento:

a) Requerimientos funcionales

Describen la funcionalidad o los servicios que se espera que el sistema proveerá. Dependen del tipo de software, del sistema que se desarrolle y de los posibles usuarios.

Cuando se expresan como requerimientos del usuario, se definen de forma general. Cuando se expresan como requerimiento del sistema, describen con detalle la función de este, sus entradas y salidas, excepciones, etc.

b) Requerimientos no funcionales

Son los requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de este, como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento.

Muchos requerimientos no funcionales se refieren al sistema como un todo más que a rasgos particulares del mismo.

A menudo son más críticos que los funcionales. Mientras que un incumplimiento de un requerimiento funcional degrada el sistema, el de un requerimiento no funcional del sistema lo inutiliza.



Los requerimientos no funcionales se clasifican según su implicancia:

- Del producto: especifican comportamiento del producto. Por ejemplo, de desempeño en la rapidez de ejecución del sistema, cuanta memoria se requiere; los de fiabilidad que fijan la tasa de fallas para el sistema sea aceptable, los de portabilidad⁵ y de usabilidad.
- Organizacionales: provienen de las políticas y procedimientos que existen en la organización del cliente y del desarrollador. Por ejemplo, los estándares que se deben utilizar en los procesos, requerimientos de implementación como los lenguajes de programación o el método de diseño a utilizar.
- Externos: cubre todos los requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo. Por ejemplo, requerimientos de interoperabilidad, requerimientos legales, requerimientos éticos.

Un problema común con los requerimientos no funcionales es que a veces son difíciles de verificar.

Idealmente, los requerimientos no funcionales se deben expresar de manera cuantitativa utilizando métricas que puedan ser probadas de forma objetiva. En la práctica, es difícil. El costo es muy alto.

c) Requerimientos del dominio

Derivan del dominio del sistema más que de las necesidades específicas del usuario.

Son importantes ya que a menudo reflejan los fundamentos del dominio de la aplicación. Si estos no se satisfacen no es posible que el sistema trabaje satisfactoriamente.

Se expresan utilizando un lenguaje específico del dominio de la aplicación que a menudo es difícil de comprender. Por ejemplo, operación para calcular desaceleración del tren, para un sistema de control de trenes.

Es importante indicar que por medio de los requerimientos se:

- Permite explicar al desarrollador cómo ha entendido lo que el cliente espera del sistema.
- Señala a los diseñadores las funcionalidades y características que tendrá el sistema resultante.
- Indica al equipo de pruebas que verificaciones se deben llevar a cabo para probar al cliente que el sistema entregado se ajusta a lo que había solicitado.

⁵ Facilidad con que un sistema software puede ser migrado entre diferentes plataformas de hardware y software.

Los requerimientos deben poseer una calidad alta para permitir una buena comprensión tanto de clientes como desarrolladores.

Con este fin, debe verificarse que los requerimientos cuenten con las características, que se obtienen de las siguientes preguntas:

- ¿Son correctos los requerimientos? Desarrollador y cliente deben revisarlos para asegurarse de que no poseen errores.
- ¿Son consistentes los requerimientos? Es decir, ¿son los requerimientos planteados no conflictivos ni ambiguos? Dos requerimientos son inconsistentes cuando es imposible satisfacerlos simultáneamente.
- ¿Son completos los requerimientos? El conjunto de requerimientos es completo si todos los estados posibles, cambios de estado, entradas, productos, restricciones están descritos en alguno de los requerimientos.
- ¿Son realistas los requerimientos? ¿Puede el sistema realmente hacer lo que el cliente está pidiendo que haga? Todos los requerimientos se deben revisar para confirmar que son factibles.
- ¿Cada requerimiento describe algo que es necesario para el cliente? Los requerimientos deben ser revisados para mantener sólo aquellos que afectan directamente en la resolución del problema del cliente.
- ¿Son verificables los requerimientos? Se deben preparar pruebas que demuestren que se han cumplido los requerimientos.
- ¿Son rastreables los requerimientos? ¿Es posible rastrear cada función del sistema hasta el conjunto de requerimientos que la establece? ¿Resulta fácil encontrar el conjunto de requerimientos que coinciden a un aspecto específico del sistema?



Fuente: Sergio Sánchez (slideshare), 2011



Tipo de requerimientos según a quién van dirigidos:

a) Requerimientos del usuario

Describen, utilizando un lenguaje sencillo que se complementa con diagramas para que sea comprensible por los usuarios sin conocimiento técnico, cuáles son las funcionalidades que esperan del sistema y las restricciones con las cuales este debe operar.

b) Requerimientos del sistema

Los requerimientos del sistema son descripciones extendidas de las funciones, servicios y restricciones que se utilizan como punto de partida para diseñar el nuevo sistema. Este documento tiene que definir con precisión lo que se implementará. Es posible incluirlo dentro del contrato entre el comprador del sistema y los desarrolladores del software.

Contextualización del análisis del problema (antecedentes)

Es importante señalar cuál es la problemática que se solucionará con la ejecución del proyecto. Los antecedentes o contextualización del problema es producto de la observación directa de la problemática y, por lo tanto, se debe realizar un breve diagnóstico, descripción, análisis y argumento del problema. Puede darse el caso de que se incluya información relevante, cualitativa y cuantitativa, de la problemática por solucionar, producto de otras investigaciones.

Generalmente el problema investigado no se presenta inmediatamente. Es normal señalar algunas situaciones o sucesos que determinan un contexto o panorama general, dentro del que aparece el problema como una situación anormal o que llama la atención ya que si es resuelto podría solucionarla o aportar a la mejora.

Para contextualizar el problema se deben contestar las siguientes preguntas:

- ¿Cómo aparece el problema que se pretende solucionar?
- ¿Por qué se origina?
- ¿Quién o qué lo origina?
- ¿Cuándo se origina?
- ¿Cuáles son las causas y efectos que produce el problema?
- ¿Dónde se origina?
- ¿Qué elementos o circunstancias lo originan?



MODELOS DE DATOS

Como se indicó anteriormente, los modelos de datos son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos. No se refiere a algo físico. Define las reglas por las cuales los datos son descritos.

Un modelo de datos permite describir de las bases de datos:

- Las estructuras de datos.
- El tipo de los datos que contiene.
- La forma en que se relacionan.

El éxito de una base de datos resulta de la combinación de dos factores:

1. La eficacia de las herramientas proporcionadas por el Sistema de Gestión de Base de Datos (SGBD).
2. El correcto diseño de la estructura de la base de datos.

Por muy potentes y adecuadas que sean dichas herramientas, el diseño es el punto clave para determinar la validez de una base de datos.


El diseño de una base de datos consiste en extraer todos los datos relevantes de un problema. Por ejemplo, saber qué datos están implicados en el proceso de facturación de una empresa que vende artículos de informática o, qué datos son necesarios para llevar el control de pruebas diagnósticas en un centro de radiológico.

Para extraer estos datos se debe realizar un análisis en profundidad del problema, para averiguar qué datos son esenciales para la base de datos y descartar los que no sean necesarios. Una vez extraídos los datos esenciales se comienzan a construir los modelos adecuados. Es decir, construir, mediante una herramienta de diseño de base de datos, un esquema que exprese con total exactitud todos los datos que el problema requiere almacenar.

Es algo equivalente al dibujo de un plano previo a la construcción de un edificio. Además, previo al diseño es necesario realizar una primera fase denominada de análisis.

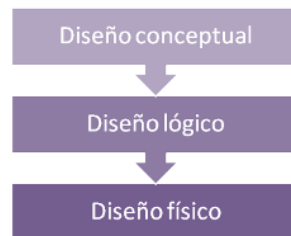
Antes de pasar a diseñar una base de datos hay que tener claro qué es lo que se quiere hacer. Para ello, tradicionalmente los informáticos se reúnen con los futuros usuarios del sistema para recopilar la información que necesitan para saber que desean dichos usuarios.

Generalmente se realiza una reunión inicial y, partir de ella, se elabora una batería de preguntas para entrevistar a los usuarios finales en una segunda reunión y obtener de ella una información detallada de lo que se espera de la base de datos.



De estas entrevistas se extrae el documento más importante del análisis: el documento de Especificación de Requisitos Software o E.R.S. A partir de dicho E.R.S, se extrae toda la información necesaria para modelar los datos.

El diseño de una base de datos es el conjunto de actividades que permite la creación de una base de datos. Esta operación se realiza en tres fases: **diseño conceptual, lógico y físico**.



Cada nivel de concreción se caracteriza por la realización de un esquema representativo de la base de datos.

- **Los modelos conceptuales.**

El modelo que se utiliza en esta primera fase del diseño tiene un gran poder expresivo para poder comunicarse con el usuario que no es experto en informática y se denomina Modelo Conceptual. El modelo conceptual que utilizaremos es el Modelo Entidad/Relación. El objetivo de esta fase del diseño consiste en representar la información obtenida del usuario final y concretarla en el E.R.S. (documento de especificación de requerimientos).

Subestimar esta etapa u obviarla por completo se traduce en bases de datos inadecuadas o ineficientes con todos los problemas que conlleva.

- **Los modelos lógicos.**

En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos. En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad, y luego se transforma el esquema conceptual en un esquema lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad.

- **Los modelos físicos.**

El modelo de datos físicos representa cómo se construirá el modelo en la base de datos. Un modelo de base de datos física muestra todas las estructuras de tabla, incluidos el nombre de columna, el tipo de datos de columna, las restricciones de columna, la clave principal, la clave externa y las relaciones entre las tablas.

Estos modelos se verán en detalle a continuación.

Tipos de modelos de datos

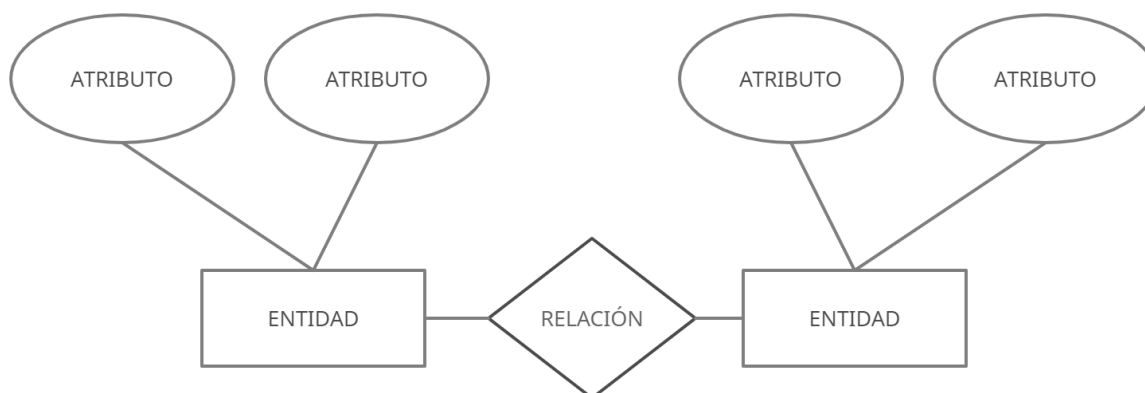
Los modelos de datos se clasifican de acuerdo con el nivel de abstracción que presentan:

Modelo conceptual

Son los orientados a la descripción de estructuras de datos y restricciones de integridad. Se usan fundamentalmente durante la etapa de Análisis de un problema dado y están orientados a representar los elementos que intervienen en ese problema y sus relaciones.

Hacen de ellos una referencia sobre un análisis que considera no pensar en lo físico ni en los procesos y si, en algo más abstracto como la estructura de los datos y cómo se pueden relacionar con otros de su misma especie.

Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción, es decir, describen los datos de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar estos datos. Mediante los modelos conceptuales se puede construir una descripción de la realidad fácil de entender.



Modelo entidad relación (solo conceptos). Fuente: elaboración propia.

Como se puede visualizar las notaciones utilizadas, estas entregan el set de símbolos necesarios para poder representar un requerimiento. Cabe mencionar que esta herramienta gráfica permite:

Representar el mundo (mini-mundo⁶) como:

- Una colección de entidades.
- Relaciones entre entidades.

Para aclarar su alcance se verá el siguiente ejemplo:

⁶ La parte de la compañía a ser representada en la base de datos.



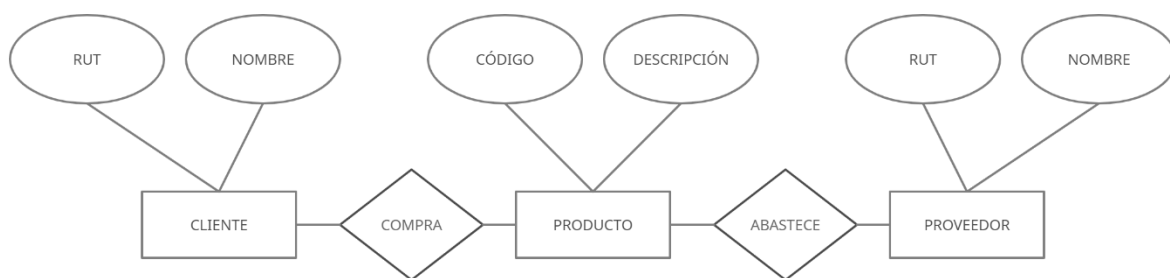
Se diseñará un modelo de datos utilizando la notación anterior el cual permitirá indicar la relación existente entre las entidades **cliente**, **producto** y **proveedor**.

Desarrollo

Se debe asignar propiedades a cada entidad para poder detallar sus características. Es importante que podamos distinguirla y asociarla a un elemento común que podamos visualizar fácilmente por medio de sus atributos.

Se debe obtener los verbos que comunicarán las entidades, dichos verbos establecerán las relaciones.

Una vez cubierto los dos puntos anteriores se genera la representación gráficamente utilizando la notación indicada.



Ejemplo de modelo entidad relación. Fuente: elaboración propia.

Modelo lógico

El propósito es obtener una representación que use de la manera más eficiente los recursos disponibles en el modelo lógico para estructurar datos y modelar restricciones.

El modelo lógico representa un sistema de manera más formal y técnica que el modelo conceptual. Describe el sistema de manera más específica y se acerca mucho más a la realidad. Este modelo puede ser menos entendible para una persona común pero no para un especialista del área.

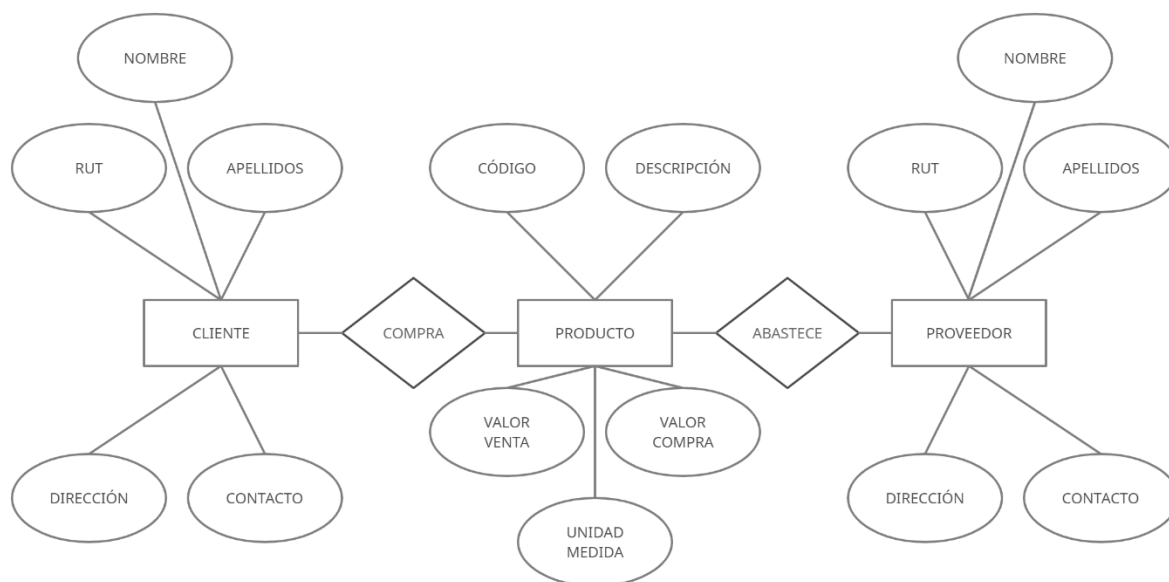
Son orientados a las operaciones más que a la descripción de una realidad. Usualmente están implementados en algún Manejador de Base de Datos. El ejemplo más típico es el Modelo Relacional, que cuenta con la particularidad de contar también con buenas características conceptuales (Normalización de bases de datos)

Un modelo lógico debe incluir:

- Todas las entidades y relaciones entre ellos.
- Todos los atributos están especificados para cada entidad.
- La clave principal de cada entidad.
- Se especifican las claves externas (claves que identifican la relación entre diferentes entidades).

El uso de llaves primarias y foráneas permite aplicar una restricción de integridad al modelo de datos.

Aplicado al ejemplo anterior se detallará el modelo lógico, haciendo hincapié en el nivel de detalle de atributos, definición de llave principal y relaciones.



Ejemplo de modelo entidad relación completo. Fuente: elaboración propia.

Cuando una entidad participa en una relación puede adquirir un papel de entidad fuerte o débil.

- Entidad débil: Una entidad débil es aquella que no puede existir sin participar en la relación, en el ejemplo anterior **producto** es este tipo de entidad ya que no puede existir sin un cliente que lo suministre.
- Entidad fuerte: Un tipo de entidad es fuerte si la existencia de sus ocurrencias no depende de ningún otro tipo. Para el caso analizado este tipo de entidad corresponde a **cliente** o **proveedor** debido que sus datos de existencia no dependen de una tercera entidad.

Un ejemplo utilizando el modelo lógico anterior es la relación entre las entidades “**proveedor**” y “**producto**”. La restricción del modelo es que, no puede existir un producto que no se encuentre asociado a un proveedor, la interpretación es que PROVEEDOR es la entidad fuerte o padre y por ende **producto** es la entidad débil o hija, si se simplifica la definición resultaría que un hijo no puede existir sin su padre, pero por otra parte el padre si puede existir sin sus hijos (Modelo Jerárquico).



Modelo físico

Se expresa haciendo uso del lenguaje de definición de datos del sistema de gestión de base de datos. El modelo de datos físicos representa cómo se construirá el modelo en la base de datos. Este modelo detalla completamente la estructura de tabla, incluidos el nombre de columna, el tipo de datos de columna, las restricciones de columna, la clave principal, la clave externa y las relaciones entre las tablas.

Las características de un modelo de datos físicos incluyen:

- Especificación de todas las tablas y columnas.
- Las claves externas se usan para identificar relaciones entre tablas.
- La desnormalización puede ocurrir según los requisitos del usuario.
- Las consideraciones físicas pueden hacer que el modelo de datos físicos sea bastante diferente del modelo de datos lógicos.

El modelo de datos físicos será diferente para diferentes Sistemas de Gestión de Base de datos. Por ejemplo, el tipo de datos para una columna puede ser diferente entre MySQL y SQL Server.

Los pasos básicos para el diseño del modelo de datos físicos son los siguientes:

- Convertir entidades en tablas.
- Convertir relaciones en claves externas.
- Convertir atributos en columnas.
- Modificar el modelo de datos físicos en función de las restricciones / requisitos físicos.



REQUERIMIENTOS INICIALES Y REALES PARA LA MEJORA DE UN MODELO DE DATOS EN UNA BASE DE DATOS

El proceso de captura de requerimientos es una fase de suma importancia dentro del proceso de desarrollo de software. Éste se preocupa de detectar y estudiar las necesidades del usuario del sistema a desarrollar.

Análisis de los requerimientos

Es el conjunto de técnicas y procedimientos que permiten conocer los elementos necesarios para definir un proyecto de software.

El análisis de requerimientos proporciona una ruta para que los clientes y los desarrolladores lleguen a un convenio sobre lo que debe hacer el sistema. La especificación, fruto de este análisis provee las pautas a seguir a los diseñadores del sistema.

Reconocer las necesidades del sistema

La identificación de los requerimientos no es una tarea sencilla ya que, el analista de sistemas tiene que recopilar información de diversas fuentes, además de encontrarse de forma constante con problemas técnicos y sociales para obtener la información.

El analista debe tener algunas características para poder alcanzar la meta de esta fase, entre ellas está la imparcialidad, la capacidad de considerar que cualquier cosa es posible, estar atento a los detalles y ser flexible, ya que cada usuario tiene una visión diferente.

Recopilación de información

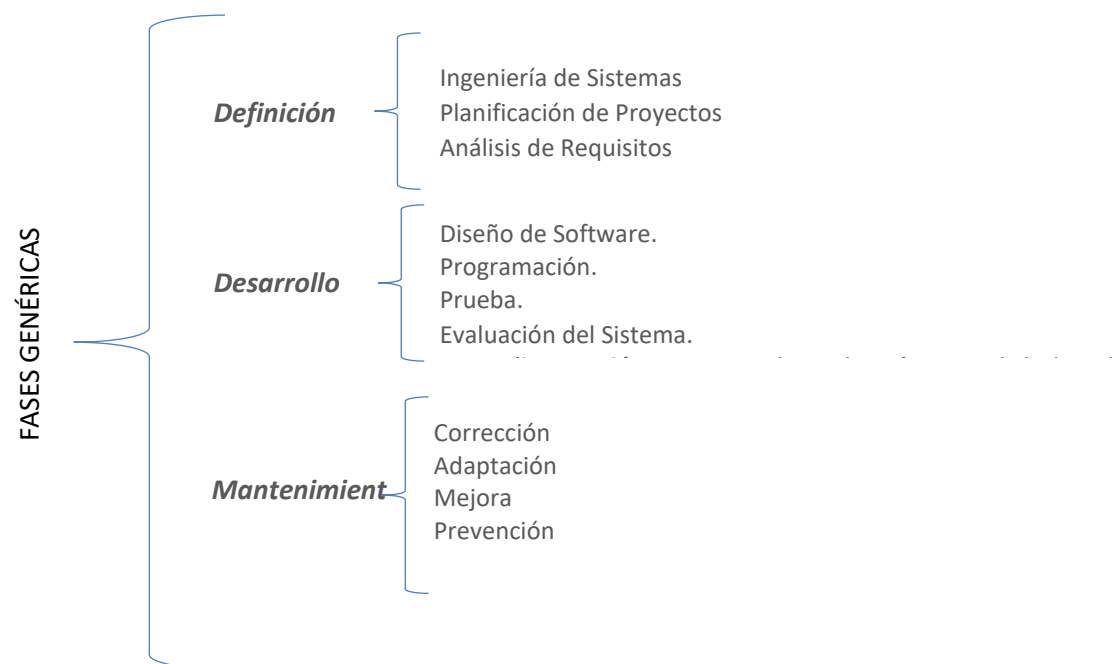
Existen diversas formas de reunir la información necesaria para poder distinguir los requerimientos. Los métodos más tradicionales consisten en las entrevistas, cuestionarios o formularios, la observación de los trabajadores y la documentación escrita de la organización.

CONCLUSIÓN

A través de los contenidos de la semana 1 has podido comprender cómo es el proceso de toma de requerimientos con los clientes y la importancia fundamental de esta etapa, ya que son la base de todo lo que se hará posteriormente. Además, conociste los procesos que siguen una vez que tenemos clara la información por parte del cliente.

IDEAS CLAVE

Cuadro sinóptico con las fases genéricas que se practican durante el proceso del software.

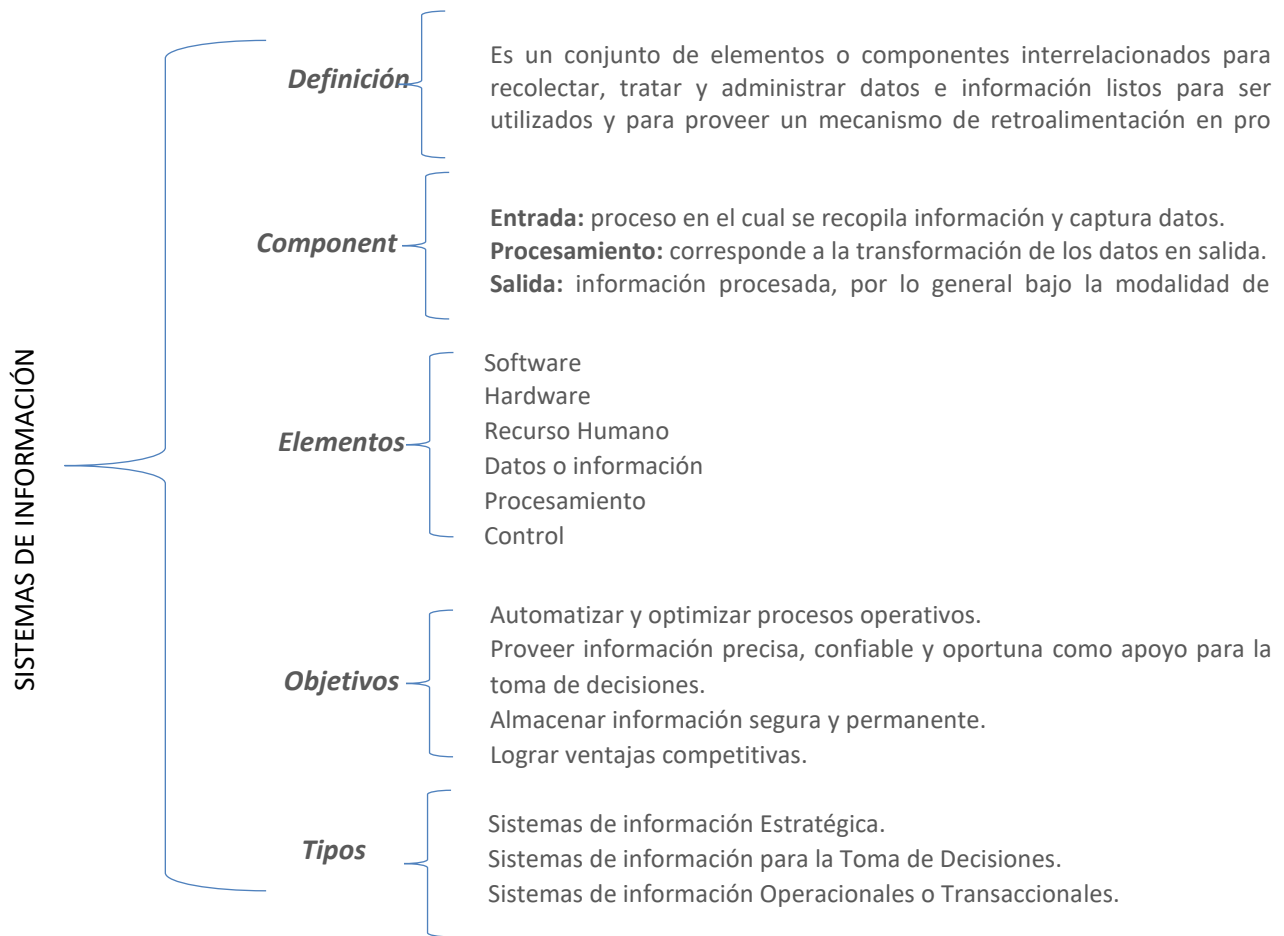


Fuente: El Informático Nica, Blog IngSoftkePC93, 2012.

Sistemas de Información

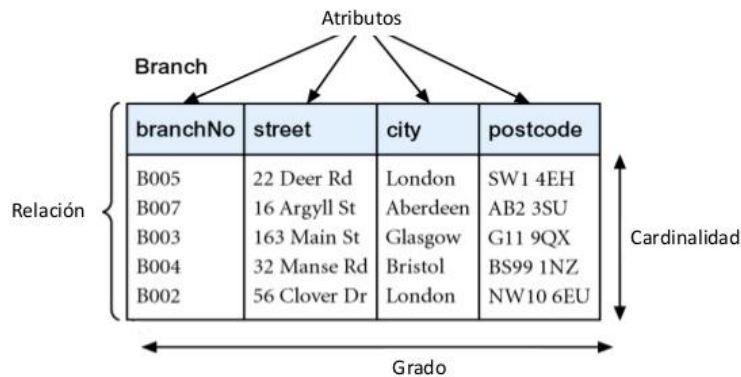
Un sistema de información se puede definir técnicamente como un conjunto de componentes relacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control en una organización.

A continuación, un cuadro sinóptico con sus fundamentos básicos:



Los sistemas de información son una poderosa herramienta que dentro de las organizaciones revoluciona el desarrollo de virtualmente todas las actividades, afectando de manera directa el desempeño de la organización dentro del mercado.

El modelo relacional



REFERENCIAS BIBLIOGRÁFICAS

Bedini González, A. P. (2005). *Gestión de Proyectos de Software*. Valparaíso: UTFSM.

Czinkota, Michael y Ronkainen, Ilkka (2008) *Marketing internacional* (8va edición). Cengage Learning.

Kendall, Kenneth y Kendall, Julie (2011) *Análisis y diseño de sistemas*. Pearson Editores.

Martínez López, F. J. y Gallegos Ruiz, A. (2017). *Programación de bases de datos relacionales*. Paracuellos de Jarama, Madrid, RA-MA Editorial.

Peña Ayala, Alejandro (2006). *Ingeniería de Software: Una Guía para Crear Sistemas de Información* (primera edición). Instituto Politécnico Nacional, México.

Stallings, William (2005). *Sistemas operativos: aspectos internos y principios de diseño* (5ª edición). Pearson Prentice Hall.