Capítulo II

Marco Teórico

Este capítulo expondrá la recopilación de antecedentes, investigaciones previas y consideraciones teóricas por donde se sustenta el presente proyecto de investigación y desarrollo, permitiendo la interpretación de los resultados y la formulación de conclusiones.

Para una mejor presentación, constara de dos partes:

En la primera parte se expone de forma detallada de los conceptos teóricos utilizados para el planteamiento del problema. Aquí se incluyen la definición de términos básicos, antecedentes, bibliografía, bases teóricas y tipos de datos.

La segunda parte se redacta la interpretación del problema bajo los términos de la teoría expuesta anteriormente.

2.1 Los sistemas de información y su importancia para las organizaciones y empresas.

Los Sistemas de Información (SI) y las Tecnologías de Información (TI) han cambiado la forma en que operan las organizaciones actuales. A través de su uso se logran importantes mejoras, pues automatizan los procesos operativos, suministran una plataforma de información necesaria para la toma de decisiones y, lo más importante, su implantación logra ventajas competitivas o reducir la ventaja de los rivales.

Las Tecnologías de la Información han sido conceptualizadas como la integración y convergencia de la computación, las telecomunicaciones y la técnica para el procesamiento de datos, donde sus principales componentes son: el factor humano, los contenidos de la información, el equipamiento, la infraestructura, el software y los mecanismos de intercambio de información, los elementos de política y regulaciones, además de los recursos financieros.

2.2 La información como recurso de las organizaciones.

La información es uno de los principales recursos que poseen las empresas actualmente. Los entes que se encargan de las tomas de decisiones han comenzado a comprender que la información no es sólo un subproducto de la conducción empresarial, sino que a la vez alimenta a los negocios y puede ser uno de los tantos factores críticos para la determinación del éxitos o fracaso de éstos.

Si deseamos maximizar la utilidad que posee nuestra información, el negocio la debe manejar de forma correcta y eficiente, tal y cómo se manejan los demás recursos existentes. Los administradores deben comprender de manera general que hay costos asociados con la producción, distribución, seguridad, almacenamiento y recuperación de toda la información que es manejada en la organización. Aunque la información se encuentra a nuestro alrededor, debemos saber que ésta no es gratis, y su uso es estrictamente estratégico para posicionar de forma ventajosala empresa dentro de un negocio.

La fácil disponibilidad que poseen las computadoras y las tecnologías de información en general, han creado una revolución informática en la sociedad y de forma particular en los negocios. El manejo de información generada por computadora difiere en forma significativa del manejo de datos producidos manualmente.

2.3 ¿ Qué es un sistema de información ?

Un Sistema de Información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. En un sentido amplio, un sistema de información no necesariamente incluye equipo electrónico (hardware). Sin embargo en la práctica se utiliza como sinónimo de "sistema de información computarizado"

Los elementos que interactúan entre sí son: el equipo computacional, el recurso humano, los datos o información fuente, programas ejecutados por las computadoras, las telecomunicaciones y los procedimientos de políticas y reglas de operación.

Un Sistema de Información realiza cuatro actividades básicas:

* Entrada de información: proceso en el cual el sistema toma los datos que requiere para procesar la información, por medio de estaciones de trabajo, teclado, código de barras, etc.
* Almacenamiento de información: es una de las actividades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardad en la sesión o proceso anterior.
* Procesamiento de la información: esta característica de los sistemas permite la transformación de los datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados o un balance general en un año base.
* Salida de información: es la capacidad de un SI para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, la voz, etc.

**Tipos de sistemas de información**

En la cultura organizacional, existen varios tipos de sistemas de información según el nivel operacional en que se utilicen. Algunos de los sistemas más comunes se encuentran a continuación:

* para procesamiento de datos (TPS: Traditional processing system): nivel operativo, destinado a procesar grandes volúmenes de información alimentando grandes bases de datos.
* sistema de expertos o basados en el conocimiento (KWS: Knowledge working systems): nivel operativo, selecciona la mejor solución para el problema presentado.
* para la administración y gerenciales (MIS: Management information systems): nivel administrativo, gestiona y elabora informes periódicos.
* para la toma de decisiones (DSS: Decision support systems): nivel estratégico, se destaca por su diseño y inteligencia que permite una adecuada selección e implementación de proyectos.
* para ejecutivos (EIS: Executive information systems): nivel estratégico, sistema personalizado para cada ejecutivo para que pueda ver y analizar datos críticos.
* sistemas funcionales relacionados con los procesos internos de la organización: forman la base de los sistemas de información para ejecutivos. Algunas de las más conocidas implementadas para las necesidades de cada área son:
  + Sistema de información de marketing (SIM)
  + Sistema de información de producción (SIP)
  + Sistema de información financiera (SIF)
  + Sistema de información de recursos humanos (SIRH)
  + Sistema de información para directivos (SDD)
  + Sistema de información geográfica (SIG)
  + Sistema de información legislativa (SIL)

**Ciclo de vida de un sistema de información**

El ciclo de vida de un sistema de información es continuo y se compone de las siguientes fases:

* Investigación preliminar, identificación de fortalezas y amenazas
* Definición de las necesidades y requerimientos
* Diseño
* Desarrollo y documentación del software
* Pruebas
* Implementación y mantenimiento
* Identificación de debilidades y oportunidades



2.4 Metodología de desarrollo de software.

Una metodología de desarrollo de software brinda al equipo de trabajo un marco para construir aplicaciones de manera eficiente y rigurosa, garantizando un producto cercano al esperado. Si no se desarrolla a partir de una metodología, el resultado final será impredecible y no se podrá controlar el avance del proyecto.

A lo largo de la historia de la computación han existido diversas metodologías, adaptadas a las posibilidades tecnológicas de cada momento:

* **Cascada**. El desarrollo es visto como el flujo de una catarata: las fases (análisis, diseño, implementación, testing, integración y mantenimiento) son secuenciales, impidiéndose iniciar una fase sin haber terminado la anterior.
* **Prototipado**. A lo largo de todo el ciclo de desarrollo se van implementando prototipos, o versiones preliminares del producto, hasta llegar al software final.
* **Incremental**. Una combinación de los anteriores. Es una metodología iterativa: en cada repetición, se entrega un producto no definitivo, pero plenamente funcional, para someterlo a la evaluación del cliente, con quien se mantiene un contacto permanente. Así, con cada iteración se avanza hacia el producto final.
* **En espiral**. Un enfoque cíclico: se repiten varias veces las etapas de planificación, análisis, evaluación y desarrollo. Se minimiza el riesgo al dividir el proyecto en segmentos pequeños y, por lo tanto, controlables.
* **Metodologías ágiles**. Esta familia de métodos es la que más se aplica en la actualidad. Todos ellos se caracterizan por una permanente interacción entre los miembros del equipo de desarrollo, poca planificación, asesoría constante por parte del cliente e iteraciones cortas, al cabo de las cuales se entrega una demostración preliminar del producto. Son tanto incrementales como iterativas.

Actualmente, la predominancia de la programación orientada a objetos y las crecientes exigencias de productividad favorecen la aplicación de las metodologías ágiles.

2.5 Modelo incremental.

El modelo incremental de gestión de proyectos tiene como objetivo un crecimiento progresivo de la funcionalidad. Es decir, el producto va evolucionando con cada una de las entregas previstas hasta que se amolda a lo requerido por el cliente o destinatario.

Este enfoque, que se usó inicialmente para proyectos de software aunque más tarde se aplicó a otros sectores, establece entregas parciales mediante un calendario de plazos. En cada una de ellas, el producto debe mostrar una evolución con respecto a la fecha anterior; nunca puede ser igual.

Una de las claves para que esto se haga efectivo es la evaluación de las etapas. Los responsables del proyecto deben analizar si los resultados parciales son los esperados y si, sobre todo, apuntan al objetivo principal. De no ser así, deberán intervenir en él e implementar las soluciones que la situación requiera.

Sin embargo, no se trata de iteraciones independientes. Por el contrario, están vinculadas de forma que cada una suponga un avance con respecto a la anterior.

**Características esenciales de este modelo son:**

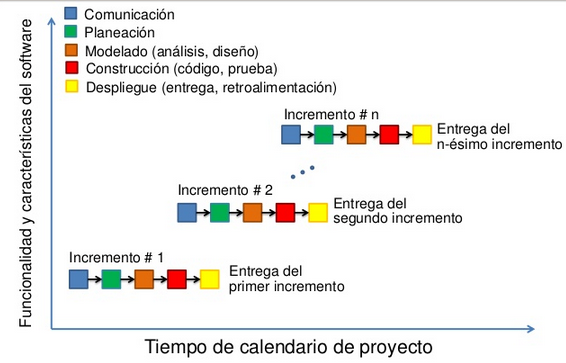
* Iteraciones dependientes, están vinculadas de forma que cada una suponga un avance con respecto a la anterior.
* Los incrementos son pequeños.
* Permite una fácil administración de las tareas en cada iteración.
* La inversión se materializa a corto plazo.
* Es un modelo propicio a cambios o modificaciones.
* Se adapta a las necesidades que surjan.

Para que esto sea posible, se debe tener en cuenta que las iteraciones no pueden ser demasiado rígidas y que no existan tareas simultáneas. El modelo incremental exige un encadenamiento progresivo de cada tarea.

**¿Cuáles son las fases del modelo incremental de gestión?**

El modelo de gestión incremental no es un modelo necesariamente rígido, es decir, que puede adaptarse a las características de cualquier tipo de proyecto, existen al menos 7 fases que debemos tener en cuenta a la hora de implementarlo:

* **Requerimientos:** son los objetivos centrales y específicos que persigue el proyecto.
* **Definición de las tareas y las iteraciones:** teniendo en cuenta lo que se busca, el siguiente paso es hacer una lista de tareas y agruparlas en las iteraciones que tendrá el proyecto. Esta agrupación no puede ser aleatoria. Cada una debe perseguir objetivos específicos que la definan como tal.
* **Diseño de los incrementos:** establecidas las iteraciones, es preciso definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe superar a la que le ha precedido. Esto es lo que se denomina incremento.
* **Desarrollo del incremento:** posteriormente se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior.
* **Validación de incrementos:** al término de cada iteración, los responsables de la gestión del proyecto deben dar por buenos los incrementos que cada una de ellas ha arrojado. Si no son los esperados o si ha habido algún retroceso, es necesario volver la vista atrás y buscar las causas de ello.
* **Integración de incrementos:** una vez son validados, los incrementos dan forma a lo que se denomina línea incremental o evolución del proyecto en su conjunto. Cada incremento ha contribuido al resultado final.
* **Entrega del producto:** cuando el producto en su conjunto ha sido validado y se confirma su correspondencia con los objetivos iniciales, se procede a su entrega final.



2.6 Patrones arquitectónicos.

Los patrones arquitectónicos se utilizan para expresar una estructura de organización base o esquema para un software. Proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos.

Los patrones arquitectónicos heredan mucha de la terminología y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos re-utilizables específicamente para la arquitectura general de los sistemas de información. En otras palabras quiere decir que a diferencia de los patrones de diseño estas son plantillas incompletas y no se pueden aplicar directamente al código con modificaciones meramente contextuales. Los patrones arquitectónicos a su vez se salen del código puro de la aplicación y suben e incluyen software, hardware, redes, inclusos las personas.

Dentro de los patrones arquitectónicos podemos encontrar:

* **Modelo Vista Controlador**: es uno de los modelos más antiguos y por lo tanto se convirtió en uno de los patrones fundamentales para el desarrollo de software. MVC a grandes trazos, separa las preocupaciones con respecto a los datos (modelo) y la interfaz de usuario (vista/GUI), permitiendo modificaciones independientes en cada una las partes sin afectar la otra, o sea, para que los cambios realizados en la interfaz de usuario (GUI) no afectan el manejo de datos, y los datos pueden ser reorganizados sin cambiar la interfaz de usuario.
* **Inyección de Dependencias:** es un patrón que a pesar de ser relativamente nuevo es muy complejo. La utilización de inyección de dependencia en un proyecto es tan incidente, que puede modificar en grandes proporciones la arquitectura, de modo que se hace prudente una planificación a futuro sobre la utilización de este patrón.
* **Arquitectura dirigida por eventos (Event-driven architecture o EDA)**: es un patrón de arquitectura software que para orquestar su comportamiento se centra en torno a la producción, detección, consumo y respuestas ante “eventos”. Teniendo en cuenta que un evento es: cualquier ocurrencia identificable que tiene un significado para el hardware o el software del sistema, en otras palabras, cualquier cambio de estado significante para el sistema. Y a su vez este cambio de estado puede ser conocido por otras aplicaciones en la arquitectura, o sea, que cada evento se propaga de manera inmediata a otras partes del sistema en la medida que sea necesario.
* **Arquitectura orientada a servicios**: es un concepto de arquitectura de software donde el software consta de una composición de servicios, prestaciones y reglas, y son los requisitos del negocio los que dictaminan la manera en la que estas se ínter-relaciona. Esta diseñado para que el sistema sea altamente escalable y flexible a nuevos requerimientos.

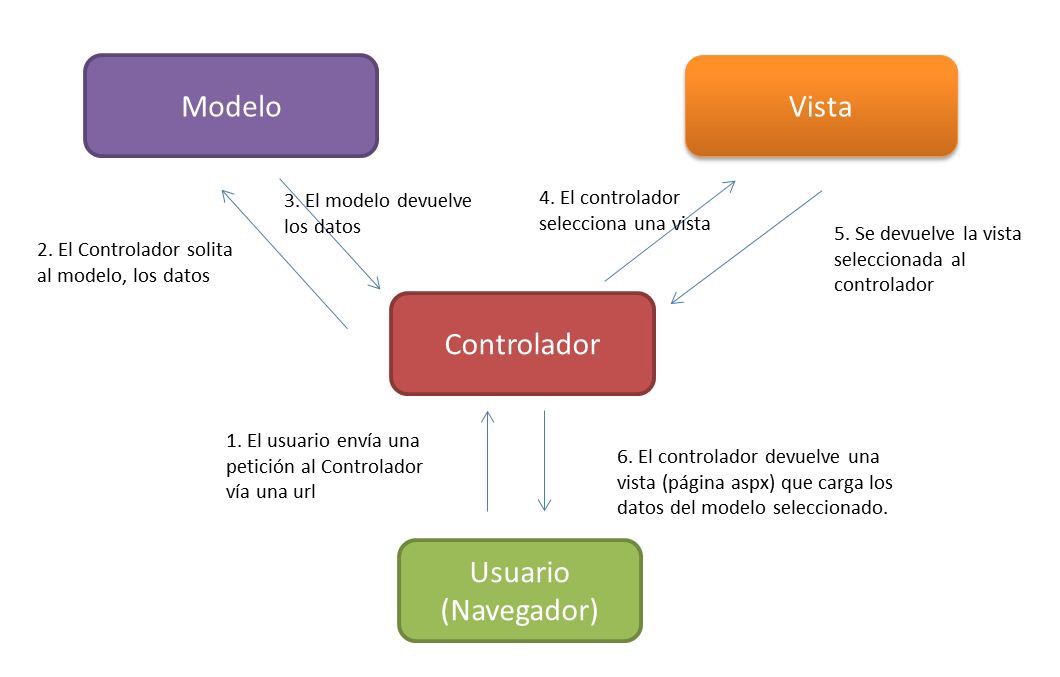
2.7 Patrón Modelo Vista Controlador (MVC).

El MVC es un patrón de diseño arquitectónico de software, que sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario. En este tipo de arquitectura existe un sistema central o controlador que gestiona las entradas y la salida del sistema, uno o varios modelos que se encargan de buscar los datos e información necesaria y una interfaz que muestra los resultados al usuario final. Es muy usado en el desarrollo web porque al tener que interactuar varios lenguajes para crear un sitio es muy fácil generar confusión entre cada componente si estos no son separados de la forma adecuada. Este patrón permite modificar cada uno de sus componentes si necesidad de afectar a los demás.

Modelo: este componente se encarga de manipular, gestionar y actualizar los datos. Si se utiliza una base de datos aquí es donde se realizan las consultas, búsquedas, filtros y actualizaciones.

Vista: este componente se encarga de mostrarle al usuario final las pantallas, ventanas, páginas y formularios; el resultado de una solicitud. Desde la perspectiva del programador este componente es el que se encarga del frontend; la programación de la interfaz de usuario si se trata de un aplicación de escritorio, o bien, la visualización de las páginas web (CSS, HTML, HTML5 y Javascript).

Controlador: este componente se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas. Por medio de él se comunican el modelo y la vista: solicitando los datos necesarios; manipulándolos para obtener los resultados; y entregándolos a la vista para que pueda mostrarlos.



2.8 Programación orientada a objetos (POO).

La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.

Está basado en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que tienen un determinado estado, comportamiento (método) e identidad:

* El estado está compuesto de datos o informaciones; serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
* El comportamiento está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
* La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta.

**Conceptos fundamentales de POO.**

* **Clase:** definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.
* **Herencia:** (por ejemplo, herencia de la clase C a la clase D) es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables publicas declaradas en C. Los componentes registrados como "privados" (private) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de POO.
* **Objeto:** Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa). Es una instancia a una clase.
* **Método:** algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.
* **Evento:** es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.
* **Atributos:** características que tiene la clase.
* **Mensaje:** una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.
* **Propiedad o atributo:** contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.
* **Estado interno:** es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

2.9 Lenguaje de programación PHP.

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje de programación considerado “del lado del servidor”, esto quiere decir que PHP se ejecuta en un servidor remoto que procesa el sitio Web antes de que éste sea abierto por el navegador del visitante. Es un lenguaje fundamentalmente pensado para el desarrollo de sitios Web dinámicos.

**Entre sus particulares principales están:**

* **Es gratuito.** Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera totalmente libre.
* **Goza de gran popularidad.** Hay una gran cantidad de programadores y desarrolladores que permanentemente realizan perfeccionamientos en su código,.
* **Tiene una enorme eficiencia.** Con muy poco mantenimiento y un servidor sin coste puede aguantar sin problema alguno millones de visitas cada día.
* **Cuenta con una simple integración con varias bases de datos**. Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos. Aunque MySQL es la base de datos que mejor trabajo con PHP puede conectarse también a otras como Oracle, o cualquier otra base de datos que sea compatible con ODBC (Open Database Connectivity Standard)
* **La versatilidad.** PHP puede utilizarse con la mayor parte de sistemas operativos, ya sea basados en UNIX como Linux, Solaris, etc. así como con Windows, el sistema operativo de Microsoft.
* **Ofrece gran cantidad de funciones predefinidas**. En contraposición de otros lenguajes para la programación, PHP fue pensado principalmente para el desarrollo de sitios Web dinámicos. Por este motivo, está provisto de una gran variedad de funciones como la descarga de documentos, el envío de mails, el trabajo con cookies y sesiones, entre otros.
* **Orientado a objetos.** partir de la versión 3 de PHP ya soportaba la programación orientada a objetos (POO), aunque es verdad que la mayoría de las características de este tipo de programación no estaban implementadas todavía.

2.10 Bases de datos.

Una base de datos es un conjunto de información perteneciente a un mismo contexto, ordenada de modo sistemático para su posterior recuperación, análisis y/o transmisión. Existen actualmente muchas formas de bases de datos, que van desde una biblioteca hasta los vastos conjuntos de datos de usuarios de una empresa de telecomunicaciones.

Las bases de datos son el producto de la necesidad humana de almacenar la información, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente. En ese sentido, la aparición de la electrónica y la computación brindó el elemento digital indispensable para almacenar enormes cantidades de datos en espacios físicos limitados, gracias a su conversión en señales eléctricas o magnéticas.

El manejo de las bases de datos se lleva mediante sistemas de gestión (llamados DBMS por sus siglas en inglés: Database Management Systems o Sistemas de Gestión de Bases de Datos), actualmente digitales y automatizados, que permiten el almacenamiento ordenado y la rápida recuperación de la información. En esta tecnología se halla el principio mismo de la informática.

En la conformación de una base de datos se pueden seguir diferentes modelos y paradigmas, cada uno dotado de características, ventajas y dificultades, haciendo énfasis en su estructura organizacional, su jerarquía, su capacidad de transmisión o de interrelación, etc. Esto se conoce como modelos de base de datos y permite el diseño y la implementación de algoritmos y otros mecanismos lógicos de gestión, según sea el caso específico.

2.11 Modelo Entidad-Relación (ER).

El modelo entidad-relación ER es un modelo de datos que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por un conjunto de objetos denominados entidades y relaciones, incorporando una representación visual conocida como diagrama entidad-relación.

**Conceptos del modelo ER**

**Ejemplares - Conjuntos - Extensión – Instancia:** Se denominan ejemplares a los registros que guardan una serie de características similares o que pueden ser agrupados o clasificados dadas sus características comunes en grupos bien delimitados. A los ejemplares también se los conoce como registros de una tabla de una base de datos, o en términos de abstracción como la extensión de la base de datos. Por ejemplo es la lista de usuarios de una biblioteca, la lista de productos con sus características, la lista de tipos de documentos y su definición.

**Entidad:** La entidad es cualquier clase de objeto o conjunto de elementos presentes o no, en un contexto determinado dado por el sistema de información o las funciones y procesos que se definen en un plan de automatización. Dicho de otra forma, las entidades las constituyen las tablas de la base de datos que permiten el almacenamiento de los ejemplares o registros del sistema, quedando recogidos bajo la denominación o título de la tabla o entidad. Por ejemplo, la entidad usuarios guarda los datos personales de los usuarios de la biblioteca, la entidad catalogo registra todos los libros catalogados, la entidad circulación todos los libros prestados y devueltos y así sucesivamente con todos los casos.

**Atributos – Intención:** Son las características, rasgos y propiedades de una entidad, que toman como valor una instancia particular. Es decir, los atributos de una tabla son en realidad sus campos descriptivos, el predicado que permite definir lo que decimos de un determinado sujeto. Por ejemplo de una entidad o tabla catálogo, se pueden determinar los atributos título, subtítulo, título paralelo, otras formas del título, autor principal, otras menciones de responsabilidad, edición, mención de edición, editorial, lugar de publicación, fecha de publicación, etc.

**Relación:** Vínculo que permite definir una dependencia entre los conjuntos de dos o más entidades. Esto es la relación entre la información contenida en los registros de varias tablas. Por ejemplo, los usuarios suelen clasificarse según una lista de tipos de usuarios, ya sean profesores, alumnos o investigadores. De esta forma es posible emitir la relación entre el usuario Jorge Martínez como alumno y Enrique Valtierra como profesor. Las relaciones son definidas de forma natural en un diagrama relacional para expresar un modelo cognitivo que dará lugar posteriormente a las interrelaciones de las entidades.

**Interrelación:** Las interrelaciones las constituyen los vínculos entre entidades, de forma tal que representan las relaciones definidas en el esquema relacional de forma efectiva. Esto no sólo la relación de los registros sino de sus tablas y de las características de la interrelación entre las entidades, a través de un campo clave que actúa como código de identificación y referencia para relacionar (es decir, como nexo de unión y articulación de la relación). Los tipos de interrelaciones entre entidades o tablas se realizan aplicando las reglas de cardinalidad y modalidad.

**Entidades fuertes:** Lo constituyen las tablas principales de la base de datos que contienen los registros principales del sistema de información y que requieren de entidades o tablas auxiliares para completar su descripción o información. Por ejemplo la tabla usuario es una entidad fuerte en relación a la tabla tipos de usuarios, que es una entidad débil dada su condición auxiliar para clasificar a los usuarios registrados en la biblioteca.

**Entidades débiles:** Son entidades débiles a las tablas auxiliares de una tabla principal a la que completan o complementan con la información de sus registros relacionados. Por ejemplo también son consideradas entidades débiles las tablas intermedias que sirven para compartir información de varias tablas principales.

**Clave:** Es el campo o atributo de una entidad o tabla que tiene como objetivo distinguir cada registro del conjunto, sirviendo sus valores como datos vinculantes de una relación entre registros de varias tablas.

**Superclave:** Es la combinación de campos clave que identifican unívocamente un registro en una tabla o entidad.

**Clave principal primaria:** Permiten identificar unívocamente cada registro de una tabla. Por ejemplo campo auto-numérico interno ID.

**Clave candidata:** Campos que cumplen las condiciones de identificación única de registros, pero que no fueron definidos como principales por el diseñador. Por ejemplo el DOI (Document Object Identifier) es un campo que define unívocamente un registro de un documento en una tabla o entidad concreta. No obstante a efectos de gestión interna del sistema el campo principal ID que contiene un valor numérico correlativo, permite un tratamiento más sencillo que el DOI.

**Clave externa:** Campo clave conformado por el valor de una clave principal primaria de otra tabla. Por ejemplo el campo id\_tipodeusuario en la tabla usuarios es un campo clave externo que guarda el valor del campo primario ID de la tabla tipodeusuario, especificando de esa forma que un usuario como Enrique Valtierra sea de tipo 2 es decir profesor.

**Integridad referencial:** Se denomina integridad referencial al tipo de interrelación que se produce entre tablas mediante un campo clave que deberá contener la cadena alfanumérica exacta al identificador de la tabla auxiliar para poder realizar la relación entre los registros. En caso contrario no se produce la relación. Además, se trata de un mecanismo que evita duplicidades e incorrecciones ya que la propiedad de integridad referencial conmina a que los datos de un usuario además de su identificador ID sean distintos al de los demás. Dicho de otra forma, no pueden existir dos registros iguales con los mismos datos.

2.12 Mysql como motor de base de datos.

MySQL, es un sistema de gestión de base de datos relacional o SGBD. Este gestor de base de datos en multihilo y multiusuario, lo que le permite ser utilizado por varias personas al mismo tiempo, e incluso, realizar varias consultas a la vez, lo que lo hace sumamente versátil.

Nació como una iniciativa de Software Libre y aún sigue ofreciéndose como tal, para usuarios particulares. Pero si se desea utilizarlo para promover datos en una empresa, se puede comprar una licencia, como un software propietario, que es autoría de la empresa patrocinante (Actualmente Oracle Corporation).

La mayor parte del código se encuentra escrito en lenguaje C/C++ y la sintaxis de su uso es bastante simple, lo que permite crear bases de datos simples o complejas con mucha facilidad. Además, es compatible con múltiples plataformas informáticas y ofrece una infinidad de aplicaciones que permiten acceder rápidamente a las sentencias del gestor de base de datos.

**Utilidad de MySQL**

Es muy utilizado en desarrollo web, ya que permite a los desarrolladores y diseñadores, realizar cambios en sus sitios de manera simple, con tan sólo cambiar un archivo, evitando tener que modificar todo el código web. Esto se debe a que MySQL, trabaja con un sistema centralizado de gestión de datos, que permite realizar cambios en un solo archivo y que se ejecuta en toda la estructura de datos que se comparte en la red. Además, permite incluir noticias e información rápidamente en un sitio web, utilizando un simple formulario, sin tener que tocar el código del website.

Cuando se combina con PHP, se convierte en una mezcla poderosa, que siempre es tomada en cuenta para realizar aplicaciones cliente/servidor, que requieran el uso de una base de datos rápida, segura y potente.

MySQL, también ofrece la posibilidad de realizar programas o aplicaciones que requieran acceso a plataformas de base de datos rápidas. Acá tiene un poco de competencia, como PostgreeSQL y otras opciones, pero al ser libre y rápido, siempre va a tener una ventaja frente a sus rivales.

2.13 PDO.

La extensión Objetos de Datos de PHP (PDO por sus siglás en inglés) define una interfaz ligera para poder acceder a bases de datos en PHP. Cada controlador de bases de datos que implemente la interfaz PDO puede exponer características específicas de la base de datos, como las funciones habituales de la extensión. Se ha de observar que no se puede realizar ninguna de las funciones de las bases de datos utilizando la extensión PDO por sí misma; se debe utilizar un controlador de PDO específico de la base de datos para tener acceso a un servidor de bases de datos.

PDO proporciona una capa de abstracción de acceso a datos, lo que significa que, independientemente de la base de datos que se esté utilizando, se emplean las mismas funciones para realizar consultas y obtener datos. PDO no proporciona una abstracción de bases de datos; no reescribe SQL ni emula características ausentes. Se debería usar una capa de abstracción totalmente desarrollada si fuera necesaria tal capacidad.

PDO viene con PHP 5.1, y está disponible como una extensión PECL para PHP 5.0; PDO requiere las características nuevas de OO del núcleo de PHP 5, por lo que no se ejecutará con versiones anteriores de PHP.

2.14 Framework.

Un framework es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Vamos, una manera de hacernos más fácil la programación. Pero… ¿Por qué debemos elegir un framework a la hora de ponernos a programar?. ¡Aquí os damos una serie de razones que te van a convencer!

**Razones para utilizar un framework a la hora de programar**

**Evitar escribir código repetitivo**

La mayoría de los proyectos tienen partes comunes necesarias para el funcionamiento como, por ejemplo, acceso a base de datos, validación de formularios o seguridad. Un framework nos evita tener que programar estas partes, de esta manera nos resulta más fácil centrarnos en programar la aplicación.

**Utilizar buenas prácticas**

Los frameworks están basados en patrones de desarrollo, normalmente MVC (Modelo-Vista-Controlador) que ayudan a separar los datos y la lógica de negocio de la interfaz con el usuario. Vamos, que gracias a ellos, lo tenemos todo más ordenado.

**Permitir hacer cosas avanzadas que tú no harías**

Está claro que un framework siempre te va permitir hacer cosas de una manera fácil y segura, que para ti serían imposibles o al menos te costaría mucho tiempo hacerlas.

**Desarrollar más rápido**

Si tenemos en cuenta los puntos anteriores, sabremos que desarrollar una aplicación con un framework nos permite hacerlo más rápido, más limpio y más seguro.

**¿Qué framework usar?**

Debemos tener claro cuáles son las características y necesidades de nuestro proyecto para elegir que tecnología utilizar. Si con un framework sencillo tenemos bastante, no hay necesidad de usar uno más complicado, ya que lo único que vamos a conseguir es complicarnos la vida. Por otro lado, utilizar el mismo framework que te fue bien en otro proyecto, tal vez no sea la decisión correcta

2.15 Progressive Web Applications (PWA).

Progressive Web Apps o PWA son referimos a un tipo de aplicaciones que podrían definirse como una página web que intenta aprovecharse de las tecnologías para ofrecer una experiencia móvil similar a una aplicación nativa. Por tanto, podríamos decir que las PWA se encuentran a medio camino entre una web y una app, puesto que intentan tomar las mejores funcionalidades de cada una.

**Ventajas Progressive Web Apps**

* Contenido fácil de rastrear e indexable
* Posibilidad de utilizar notificaciones push.
* Optimización del rendimiento en dispositivos móviles.
* Velocidad de carga totalmente optimizada
* Ofrece la posibilidad de incluir en el escritorio o menú de un dispositivo mobile, un icono que nos lleva directamente a la PWA.
* Podemos acceder al portal sin necesidad de disponer de conexión.
* Uso del HTTPS para todas las solicitudes.

**¿Cómo funciona una Progressive Web App (PWA)?**

Uno de los elementos más importantes que conforman las PWA es el manifiesto de la aplicación y que está en formato JSON. Este, nos permite controlar cómo se verá nuestra aplicación en áreas donde normalmente los usuarios tienen aplicaciones nativas.

Este archivo o manifiesto contiene lo siguiente:

* Name: Nombre de la aplicación que aparecerá en el menú mobile del usuario.
* Description: Indicar la descripción de nuestra aplicación móvil
* Icons: Crear distintos iconos, con resoluciones distintas, para que de esta manera, se vea bien en todos los dispositivos.
* Start url: URL de inicio cuando abrimos la aplicación
* Display: En este caso, podemos elegir varias configuraciones (standalone, fullscreen, minimal-ui, entre otros)
* Orientation: Cuando hablamos de orientación nos referimos a si queremos que la aplicación web se deba usar en modo retrato o en modo paisaje.
* Theme\_color: el color que se usará para la barra superior de la aplicación.
* Background\_color: color para la pantalla antes de la carga completa de la aplicación.

Otro elemento que encontramos es el Service Workers, se trata de uno o varios scripts que funcionan en un segundo plano y que nos permiten generar una experiencia offline, descargando los contenidos imprescindibles para que la aplicación funcione, así como generar notificaciones push.

El **Service Workers** es realmente importante ya que nos permite optimizar la retención de los usuarios. Hasta la fecha, esta funcionalidad solo la tenían las aplicaciones nativas, pero se ha convertido en una de las funcionalidades más importantes para poder mejorar el retorno del usuario a nuestra app. No obstante, con el aumento de notificaciones en todas las app, esta funcionalidad o característica cada vez queda más oculta.

**Elementos más importantes del DOM de una Progressive Web App**

Las Progressive Web App se componen de dos partes principales, el App-Shell, que hace referencia a la estructura de la página y el contenido, que puede variar entre las distintas páginas de nuestra aplicación.

**Medir la velocidad de una PWA**

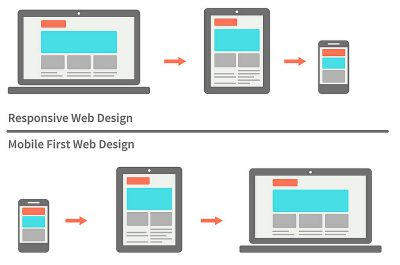
El principal beneficio de las PWA son su peso, se trata de páginas que completas pueden tener un peso de menos de 1MB, ofreciendo una experiencia veloz para los usuarios. Esto aporta una gran ventaja al mundo mobile-first al que nos encontramos.

2.16 Mobile first.

El modelo Mobile First es una filosofía que perfecciona el proceso de diseño de cualquier sitio web, buscando mejorar el diseño web adaptable. Esta nueva forma de diseñar, surge de los dispositivos móviles y de cualquier pantalla de tamaño menor, en la que se parte de una escala en aumento hasta adaptar el diseño a pantallas de gran tamaño.

Para aplicar esta filosofía, debemos tener en cuenta que todo lo que no sea esencial debe quedar fuera de nuestro diseño. Para hacerlo debemos pensar en lo que buscan nuestros consumidores, qué es lo que no puede faltar en ningún caso y qué es prescindible o superfluo.

Cuando se aplica el Responsive Web Design, cada vez es más habitual que se tomen las decisiones pensando en este método. Esto se debe a que la mayoría de los usuarios actualmente acceden a la información a través de sus dispositivos móviles.



**Aspectos a tener en cuenta en el diseño Mobile First**

Teniendo en cuenta esta manera de enfocar el diseño, las dimensiones y capacidades de un móvil y la información que debemos incluir, podemos pasar al desarrollo de una página web.

Lo primero que se debe hacer es aclarar el contenido básico y separarlo del contenido sobrante o prescindible. Al hacer esto, nos quedará un listado de los elementos esenciales que debe contener nuestra página; home, página interior, carrito de compra, buscador… Estos elementos deben decidirse en función de las necesidades de los futuros usuarios.

Puede darse el caso de que ciertos elementos no se logren incluir a la vez en pantalla como el logotipo, el navegador y el buscador, por ejemplo. Si el espacio realmente es ínfimo, siempre puede darse la opción de habilitar un botón con categorías para las opciones que no se puedan incluir en la pantalla.

**Diseño**

En la parte de diseño el proceso empieza igual, debemos pensar en cómo veríamos la página web si entráramos desde un móvil. A partir de ahí, se diseña la interfaz de la página, tanto en papel durante el proceso de prototipado como en HTML + CSS en la práctica y desarrollo.

Es mucho más sencillo diseñar una página web partiendo de un dispositivo móvil que desde un ordenador de escritorio, sobre todo por las dimensiones y la cantidad de elementos que pueden incluirse en uno u otro. Trabajar con un espacio reducido es realmente productivo y eficiente, se consigue disminuir la información y se mantiene lo importante, desechando de esta forma lo que es secundario.

Una vez obtenemos el diseño definitivo para un dispositivo de tamaño reducido, el siguiente paso es estirar la ventana hasta que el diseño empiece a deformarse o a verse mal. Este es justo el momento de usar lo que llamamos ‘breakpoint’ (el punto de ruptura) y definir los media queries para pantallas de mayor dimensión.

Mobile First hace más sencilla la tarea de diseñar cualquier página web, de manera gradual el diseño se irá acomodando a los distintos tamaños de pantalla, adaptando el layout al tiempo.

Diseñar a partir de pantallas grandes es mucho menos práctico, requiere mucho más esfuerzo reducir que ir aumentando el espacio. Los diseñadores suelen estar de acuerdo en esto, lo mejor es empezar diseñando para pantallas móviles.

2.17 Responsive Design.

Responsive Design o diseño adaptativo, es la técnica que se usa en la actualidad para tener una misma web adaptada a las diferentes plataformas que nos brinda la tecnología: ordenador, tablet y Smartphone.

Consiste en una serie de hojas de estilo en CSS3, que usando el atributo “mediaquery” convierten una web ordinaria en una web multiplataforma capaz de adaptarse a todos los tamaños que existen, ofreciendo una experiencia para el usuario mucho más amena y cubriendo las necesidades de nuestro público. Atrás quedan las webs que vistas en un Smartphone se tenían que ampliar para poder leer algo.

**Ventajas de ser Responsive**

* **Mejor experiencia de usuario.** Eso repercute en la opinión que los usuarios tienen de tu sitio web y el uso que le darán a ella. Mejorará tanto tu imagen de marca como el tiempo de permanencia en la web y aumentará la tasa de rebote entre páginas de tu web.
* **Se acabaron los contenidos duplicados.** Si usabas una versión móvil para que los usuarios que accedían a tu web vieran el contenido adaptado al dispositivo, era perjudicial para tu SEO, ya que eso creaba contenido duplicado y por lo tanto Google penalizaba a tu web. Un diseño Responsive evita este problema ya que es el mismo contenido que se organiza de manera distinta según el dispositivo en el que se vea.
* **Se reducen los costes de desarrollo y mantenimiento de la web.** Al usar la misma plantilla para todas las plataformas, se reduce la inversión en mantenimiento y desarrollo, ya que un sólo cambio repercute en todas las versiones.
* **Aumenta la viralidad.** Un reciente estudio certifica que el 70% de los accesos a las redes sociales se hacen a través de dispositivos móviles, eso significa que si un usuario accede a nuestra web a través de un Smartphone, seguramente tenga las aplicaciones sociales abiertas y si quiere compartir algo, es mucho más rápido y natural.

2.18 UI/UX

**UI, User Interface (Interfaz de Usuario)**

La UI (por sus siglas en inglés User Interface) o en español Interfaz del Usuario, es la vista que permite a un usuario interactuar de manera efectiva con un sistema. Es la suma de una arquitectura de información + patrones de interacción + elementos visuales.

Laurel y Mountford la definen como:

“La superficie de contacto que refleja las propiedades físicas de los interactores, las funciones que pueden realizarse, y el balance entre poder y control.”

En otras palabras:

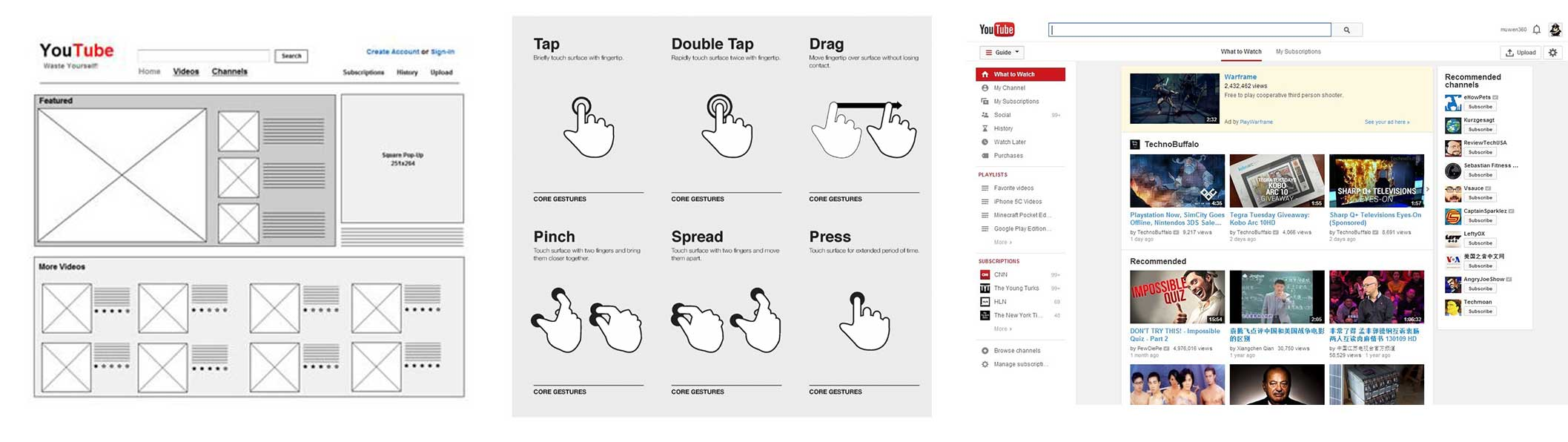
“El lugar donde el contacto entre dos entidades ocurre.”

De manera breve las definiría como:

“La representación visual, táctil y/o fonética que permite a un usuario interactuar con un sistema.”

**Componentes de la UI (Interfaz de Usuario)**

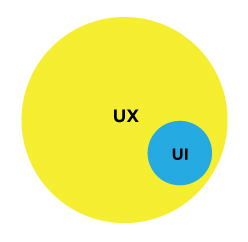
La UI, principalmente presenta las posibilidades de interacción junto con el look & feel, o en otras palabras, el diseño visual. Esto quiere decir, que las decisiones de Arquitectura de Información, Diseño de Interacción y/o Diseño Visual se ven reflejadas aquí. Mira el ejemplo a continuación:



La arquitectura dicta la estructura y el esqueleto, además se piensa en cómo se comporta el sistema en respuesta al usuario y por último tiene una capa de colores, texturas y gráficos que lo hacen estético.

Según Usability.gov, los elementos que generalmente se presentan en una UI para un producto digital son:

* **Controles de Input:** botones, campos de texto, checkboxes, radio buttons, listas drop down, switches, campos de fecha
* **Componentes de Navegación:** breadcrumbs, sliders, formularios de búsqueda, paginación, sliders, tags, iconos
* **Componentes de Información:** tooltips, iconos, barras de progreso, notificaciones, cajas de mensajes, modals,
* **Contenedores:** acordeones

La UI influye en la generación de una UX

**Principios del Diseño de UI (Interfaz de Usuario)**

Según Larry Constantine y Lucy Lockwood, existen ciertos principios para el diseño de interfaz centrado en el usuario. Estos principios son:

* **Estructura:** El diseño debe organizar la interfaz de usuario a propósito, de manera significativa y útil basada en modelos claros y consistentes que son evidentes y reconocible para los usuarios, poniendo cosas relacionadas entre sí y que separa las cosas no relacionadas, diferenciando cosas diferentes y hacer cosas similares se parecen entre una y otra. El principio de estructura concierne a la arquitectura general de la interfaz de usuario.
* **Simplicidad:** El diseño debe facilitar las tareas sencillas y comunes, comunicarse de forma clara y sencilla en el propio idioma del usuario y proporcionar buenos accesos directos que estén significativamente relacionados con procedimientos más largos.
* **Visibilidad:** El diseño debe hacer visibles todas las opciones y materiales necesarios para una tarea determinada sin distraer al usuario con información superflua o redundante. El buen diseño no abruma a los usuarios con alternativas o confunde con información innecesaria.
* **Retroalimentación:** El diseño debe mantener a los usuarios informados de las acciones o interpretaciones, los cambios de estado o condición, y los errores o excepciones que son relevantes y de interés para el usuario a través de un lenguaje claro, conciso y claro que sea familiar para los usuarios.
* **Tolerancia:** El diseño debe ser flexible y tolerante, reduciendo el costo de los errores y el mal uso al permitir deshacer y rehacer, y al mismo tiempo evitar los errores siempre que sea posible, tolerando entradas y secuencias variadas e interpretando todas las acciones razonables.
* **Reuso:** El diseño debe reutilizar los componentes y comportamientos internos y externos, manteniendo la coherencia con el objetivo en lugar de consistencia meramente arbitraria, reduciendo así la necesidad de los usuarios a repensar y recordar.

**UX, User Experience (Experiencia de Usuario)**

La UXPA define UX (Experiencia de Usuario) como:

“Cada aspecto de la interacción del usuario con un producto, servicio o empresa conforman las percepciones del usuario.”

Para Nielsen Norman Group:

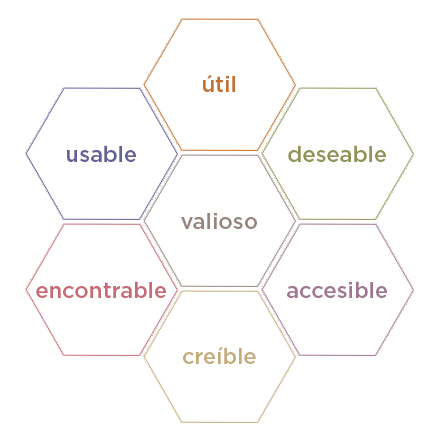
“La experiencia del usuario abarca todos los aspectos de la interacción del usuario final con la empresa, sus servicios y sus productos.”

No hay una definición exacta, pero en simple palabras:

“UX es aquello que una persona percibe al interactuar con un producto o servicio.”

**Facetas de la UX (Experiencia de Usuario)**

Peter Morville muy conocido por el libro “del oso polar” define muy bien las facetas que hay que tomar en cuenta para diseñar una buena Experiencia de Usuario. Morville diseñó este diagrama para poder transmitir a sus clientes que además de la Usabilidad, era importante tomar en cuenta otros aspectos. El diagrama es conocido como “UX Honeycomb”.

* **Útil:** Como profesionales no podemos quedarnos sólo con seguir las instrucciones de clientes o jefes. Debemos de ser capaces de descubrir y crear productos que realmente resuelvan un problema o dolor.
* **Usable:** Es cierto que la usabilidad, que se refiere a qué tan fácil es de usar un sistema, sigue siendo importante, pero no es lo único.
* **Deseable:** Los productos atractivos funcionan mejor como lo menciona Don Norman en su libro El Diseño Emocional.
* **Encontrable:** Debemos enfocarnos en crear sistemas donde el usuario pueda navegar fácilmente y encontrar la información que busca.
* **Accesible:** Más de 10% de la población tiene algún tipo de discapacidad (auditiva, visual, motriz…) Además de ser ético, es un aspecto legal que hay que tomar en cuenta.
* **Creíble:** ¿Qué elementos causan una buena impresión y crean confianza en los usuarios? Debemos de diseñar para la credibilidad.
* **Valioso:** Nuestros productos deben de cumplir la misión de la organización, pero además ofrecer un valor agregado y mejorar la satisfacción del usuario.