2024

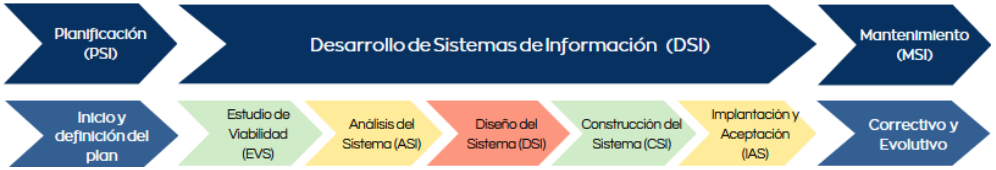
Bloque III. Desarrollo de Sistemas

TÉCNICO AUXILIAR INFORMÁTICO

Índice

Bloque 3 - Tema 1: MODELO CONCEPTUAL DE DATOS

MÉTRICA versión 3 es Metodología de Planificación, Desarrollo y Mantenimiento de SI.
Paradigmas de desarrollo (Estructurado y Orientado a objetos).
El enfoque principal consiste en descomponer cada uno de los procesos en actividades, y éstas a su vez en tareas. PROCESOS > ACTIVIDADES > TAREAS



Proporciona también cuatro **interfaces** des orientadas a la mejora de los procesos principales:
Gestión de Proyectos, Seguridad MAGERIT, Aseguramiento de Calidad y Gestión de la Configuración.

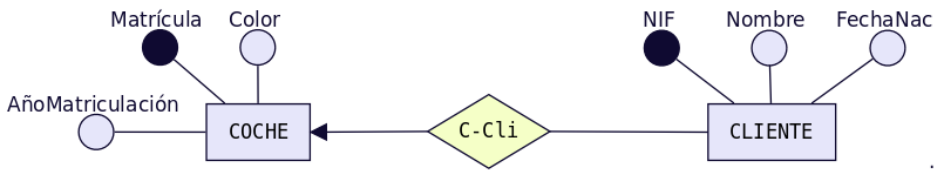
Perfiles participantes en el desarrollo:		
Directivo – Comité de Dirección – Comité de Seguimiento – Directores de usuarios – Usuarios expertos	**Jefe de proyecto** – Responsable de Implantación – Resp. de Mantenimiento – Resp. de Operación – Resp. de Sistemas – Resp. de Seguridad – Resp. de Calidad	**Consultor** – Consultor Informático – Consultor de las TI – Consultor de SI – Especialista en Comunicaciones – Técnico de Sistemas – Técnicos de Comunicaciones
Analista – Administrador de BBDD – Equipo de Arquitectura – Equipo de Formación – Equipo de Implantación – Equipo de Operación – Equipo de Seguridad – Equipo de Soporte Técnico – Equipo de Proyecto – Grupo de Aseg. de la Calidad	El perfil de **Programador** hace referencia únicamente al participante Programador descrito en MÉTRICA Versión 3	

Los tipos de pruebas que deben realizarse son:

- **Pruebas unitarias:** Verificar la funcionalidad de cada componente individualmente una vez que ha sido codificado. (Enfoque caja blanca y caja negra).
- **De integración:** Verificar el correcto ensamblaje entre los distintos componentes con el fin de comprobar que interactúan correctamente a través de sus interfaces. (Estrategias top-down, bottom-up, combinadas).
- **Del sistema:** Ejercitar profundamente el sistema comprobando la integración del sistema globalmente, verificando los distintos subsistemas con el resto de sistemas (Tipos: funcionales, comunicaciones, rendimiento, volumen, sobrecarga, disponibilidad de datos, facilidad de uso, operación, entorno y seguridad.
- **De implantación:** Comprobar el funcionamiento correcto del sistema integrado de hardware y software en el entorno de operación, así como la aceptación del sistema una vez instalado en su entorno real. (Gestión de copias de seguridad y recuperación, pruebas de sobrecarga o de stress).
- **De aceptación:** Validar que un sistema cumple con el funcionamiento esperado y permitir su aceptación, desde el punto de vista de su funcionalidad y rendimiento. (Pruebas de caja negra que demuestran la conformidad con los requisitos).
- **De regresión:** Eliminar el efecto onda, es decir, comprobar que los cambios sobre un componente no introducen un comportamiento no deseado o errores adicionales. (implica la repetición de las pruebas que ya se han realizado previamente).

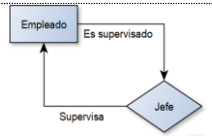
Modelo E/R: técnica (reglada) para la representación gráfica del **modelo conceptual** que ayuda su comprensión, detección previa de posibles errores y la mejora de su mantenimiento. Es propuesta por MÉTRICA v3 creado por Peter Chen en 1.976.

Modelo E/R: ELEMENTOS BÁSICOS	
Dominio: Conjunto de valores homogéneos (tipo de dato). El dominio tiene existencia propia.	Entidad: objeto u elemento (real o abstracto) con características diferenciadoras capaces de hacerse distinguir de otros objetos.
Atributo: Propiedades o características que tiene un tipo de entidad o un tipo de relación se denomina atributo; los atributos toman valores de uno o varios dominios.	Relación: correspondencia que existe entre un o más entidades. Las relaciones pueden ser regulares si asocia con entidades regulares o débiles si asocia entidades débiles.

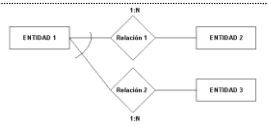


Modelo E/R: TIPOS DE RELACIONES

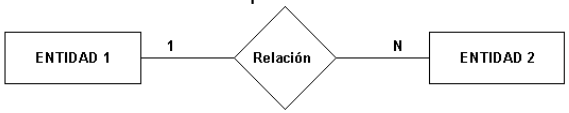
Reflexiva: cuando una entidad está relacionada consigo misma.



Exclusivas: se incluye un arco sobre las líneas que conectan el tipo de entidad a los dos o más tipos de relación.



Correspondencia: número máximo de ocurrencias de cada tipo de entidad que pueden intervenir en una ocurrencia de la relación que se está tratando.



Cardinalidad: es el número máximo y mínimo (min, max) de ocurrencias de un tipo de entidad que pueden estar interrelacionadas con una ocurrencia de otro tipo de entidad. La cardinalidad máxima coincide con el tipo de correspondencia. MAX=1,n

- Obligatoria, cuando para toda ocurrencia de un tipo de entidad existe al menos una ocurrencia del tipo de entidad asociado (1, max).
- Opcional, cuando para toda ocurrencia de un tipo de entidad, puede existir o no una o varias ocurrencias del tipo de entidad asociado (0, max).

Generalización: abstraer un tipo de entidad de nivel superior (supertipo) a partir de varios tipos de entidad (subtipos).

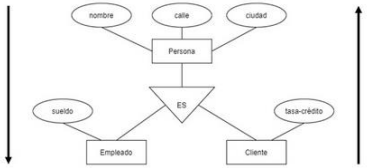
Especialización: un supertipo se descompone en uno o varios subtipos, los cuales heredan todos los atributos y relaciones del supertipo, además de tener los suyos propios.

Categoría: es el subtipo que aparece como resultado de la unión de varios tipos de entidad.

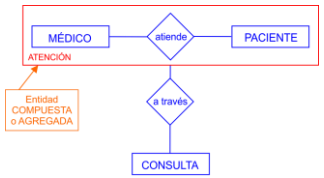
Agregación: construir un nuevo tipo de entidad como composición de otros y su tipo de relación.

Asociación: relacionar dos tipos de entidades que normalmente son de dominios independientes, pero coyunturalmente se asocian.

Especialización



Generalización



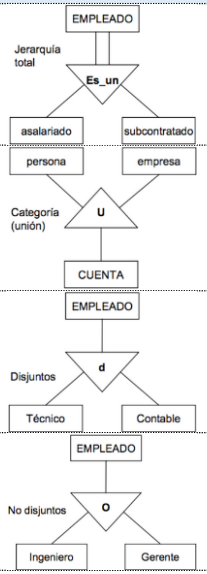
Modelo E/R: JERARQUÍAS

TOTAL: toda ocurrencia del supertipo pertenece siempre a algún subtipo.

PARCIAL: una ocurrencia del supertipo no pertenece a algún subtipo.

DISJUNTA: una ocurrencia del supertipo solo pued pertenecer a uno de los subtipos.

SOLAPADA: una ocurrencia del supertipo puede pertenecer a varios subtipos.

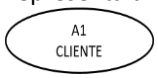


Se puede mezclar cualquiera de las opciones {total | parcial} con cualquier otra de las siguientes {disjunta | solapada}.

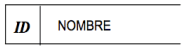
Diagrama de Flujo de Datos (DFD): su objetivo es la obtención de un modelo lógico de procesos que represente el sistema, con independencia de las restricciones físicas del entorno.

Diagrama de Flujo de Datos (DFD): ELEMENTOS BÁSICOS

Entidad externa: Representa un ente ajeno.

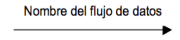


Almacén de datos: Representa la información en reposo.



Proceso: Representa una funcionalidad.

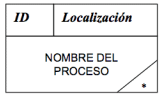
Flujo de datos: Representa el movimiento de los datos.



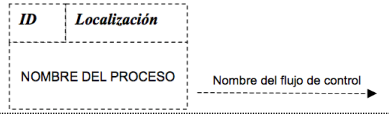
Si aparece varias veces una entidad externa:

Si aparece varias veces un almacén:

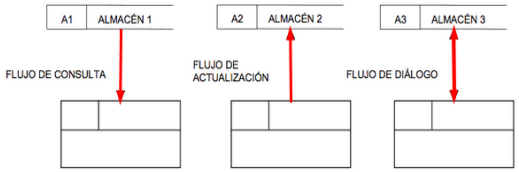
Si el proceso es de último nivel:



Proceso (coordina) y flujo de control (tiene como destino un proceso de control):



Tipos de flujos de datos:



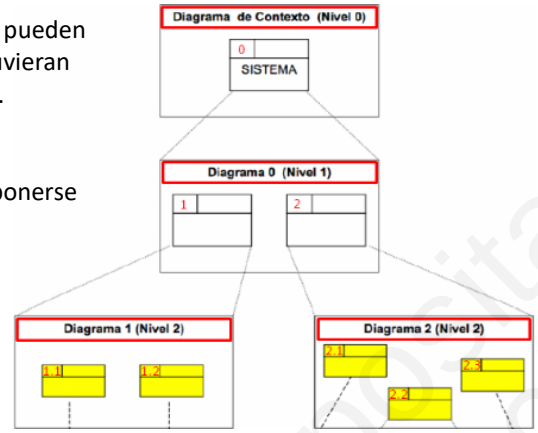
Diálogo: representa una consulta y una actualización.
La conexión entre Entidad externa – Entidad externa está prohibida en los DFD.

La descomposición por niveles se realiza de arriba abajo (top-down), es decir, se comienza en el nivel más general y se termina en el más detallado

En un DFD resultado de una explosión no pueden aparecer almacenes de datos que no estuvieran ya asociados al proceso del nivel superior.

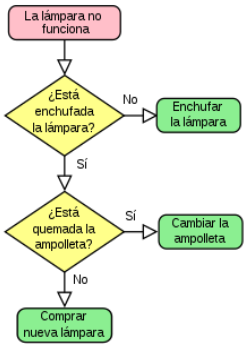
A los procesos que no necesitan descomponerse Se les denomina Procesos primitivos.

Si en el nivel 2 hay 5 diagramas significa que se han identificado 5 subsistemas.



El diagrama de flujo o flujograma: es la representación gráfica de un algoritmo o proceso.
Son utilizados para la definición, análisis o solución de problemas.

Símbolos utilizados	Nombre	Explicación
	Línea de Flujo	Muestra la dirección del sentido del flujo del proceso conectando los símbolos.
	Inicio/Fin	Utilizado para iniciar un proceso o para terminarlo.
	Datos Entrada/Salida	Los datos para realizar una actividad.
	Proceso	Tarea o Actividad llevada durante el proceso.
	Decisión	Indicamos puntos en que se toman decisiones.
	Imprimir Documento	Generación o consulta de documento específico.
	Conector	Hace conexión dentro de una misma página.
	Conector de Página	Conecta con otra página.
	Pantalla	Imprime un mensaje o resultado en pantalla.



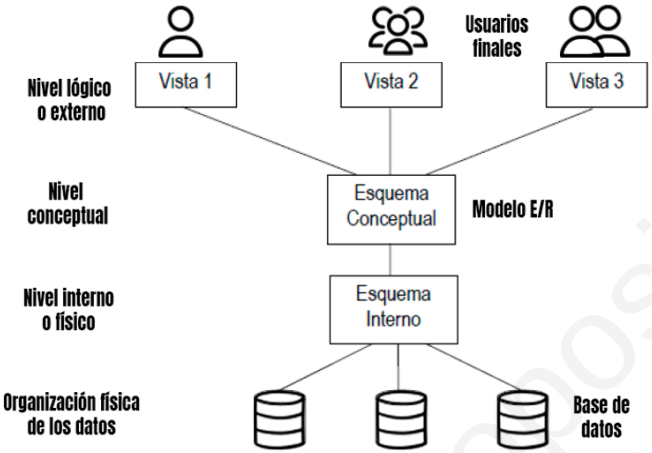
Bloque 3 - Tema 2: DISEÑO DB. EL MODELO LÓGICO RELACIONAL. NORMALIZACIÓN

Modelo ANSI/SPARC: Las bases de datos deben tener la capacidad de adaptación a cambios en el entorno. Conseguir la independencia de datos (independencia lógica y física) tratamos de descomponer el proceso en tres capas o niveles:

Nivel lógico o externo: se refiere a las vistas o una porción de la Base de Datos completa que se muestras a los usuarios o aplicaciones.

Nivel conceptual: en este nivel se describe la estructura y las relaciones que existen entre datos. En este nivel se realiza la adecuación del esquema conceptual al modelo lógico.

Nivel interno o físico: Cómo y dónde se almacenan físicamente los datos, se describen estructuras de datos complejas.



Modelo Lógico Relacional: modelo con fundamentos matemáticos basado en la teoría de conjuntos. Definido en 1970 por Edgar Frank Codd. Objetivos:

Independencia Física: La forma de almacenar los datos no debe influir en su manipulación. Si el almacenamiento físico cambia, no se tienen que modificar sus aplicaciones.

Independencia Lógica: Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se inserten, actualicen y eliminen datos.

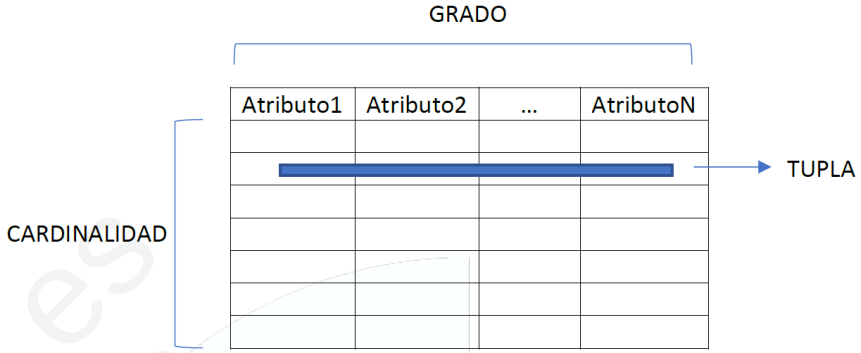
Flexibilidad: Poder presentar a cada usuario los datos de la forma en que éste prefiera

Uniformidad: Las estructuras lógicas de los datos siempre tienen una única forma conceptual (las tablas), lo que facilita la creación y manipulación de la base de datos por parte de los usuarios.

Sencillez: Las características anteriores hacen que este Modelo sea fácil de comprender y de utilizar por parte del usuario final.

Conceptos fundamentales:

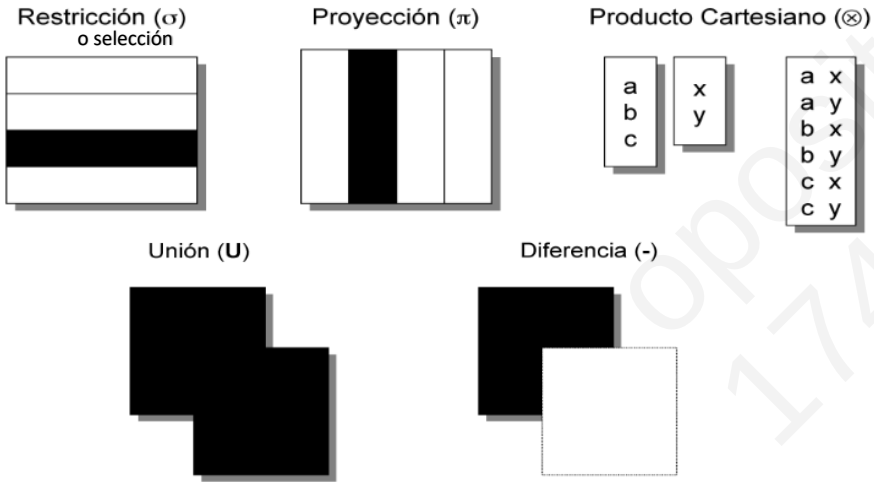
- Un dominio en un conjunto de valores.
- El número de atributos de una relación define su grado.
- El número de tuplas de la relación define su cardinalidad.



Paso del modelo E/R al Modelo Relacional	
Relaciones 1:1 **Se propaga la clave** en las dos direcciones.	Relaciones 1:N **Se propaga la clave** en sólo UNA de las dos direcciones.
Relaciones N:M y ternarias **Se crea una tabla nueva** como consecuencia de la relación que tendrá como clave primaria la concatenación de los identificadores de las entidades relacionadas. La condición de clave ajena se expresa mediante FOREIGN KEY.	Relaciones Reflexivas Reflexivas con correspondencia 1:N se transforma con **propagación de clave**, aunque, al tratarse de la misma tabla, es necesario renombrar el nombre del identificador que se transfiere, ya que si no estaría repetido respecto del ya existente en la entidad de origen.
Reglas de integridad	
Regla de integridad de la **ENTIDAD**: Se aplica a las claves primarias de las relaciones Ninguno de los atributos que componen la clave primaria puede ser nulo.	Regla de integridad **REFERENCIAL**: Si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o ser completamente nulos.

Normalización:
1FN -> si no tiene grupos repetitivos, es decir, un atributo sólo puede tomar un único valor de un dominio simple. **Si los dominios de los atributos son atómicos**.
2FN -> si está en 1FN y todos los atributos que no forman parte de las claves candidatas (atributos no principales) **tienen dependencia funcional completa** respecto de éstas.
3FN -> si está en 2FN y todos sus atributos no principales dependen directamente de alguna de las claves, es decir, **no hay dependencias funcionales transitivas** de atributos no principales respecto de las claves.
FN de Boyce-Codd (**FNBC**) -> Si está en 3FN y además **todo determinante es una clave candidata**.
4FN -> Si está en FNBC y todas las dependencias múltiplemente valoradas el **implicante es clave candidata.**
5FN -> Si está en 4FN y toda dependencia de combinación está implicada por una clave candidata.

Álgebra Relacional: hay 5 fundamentales, que forman un conjunto relacionalmente completo, es decir, permite obtener cualquier subconjunto de los datos contenidos en una BD



No fundamentales:
JOIN o concatenación: relación resultante de la **selección, proyección y producto cartesiano.**
INTERSECCIÓN: relación resultante de tomar las filas que están tanto en R como en S.
DIVISIÓN: dadas dos relaciones R y S en las que existe un subconjunto de atributos (X) que están formando parte de S y R al mismo tiempo y otro conjunto de atributos (Y) que únicamente forman parte de R, se define la división o cociente como la relación resultante de combinar cada fila de Y con todas las filas que forman parte de los atributos X.

Diseño físico: se divide de cuatro fases, cada una compuesta por una serie de pasos:
- Traducir el esquema lógico para el SGBD específico mediante lenguaje DDL.
- Diseñar la representación física (productividad transacciones, tiempo respuesta y espacio disco).
- Diseñar los mecanismos de seguridad mediante lenguaje DCL.
- Monitorizar y afinar el sistema.

La optimización consiste en una **desnormalización** controlada **del modelo físico** de datos que se aplica para reducir o simplificar el número de accesos a la base de datos.

Bloque 3 - Tema 3: LENGUAJES DE PROGRAMACIÓN

Definiciones básicas

Algoritmo: conjunto de acciones o secuencia de operaciones que resuelven un problema concreto. Existen n algoritmos para resolver un mismo problema, escoger el más eficiente.

Instrucción: cada una de las tareas elementales a realizar por un elemento de computación.

Programa: es un conjunto de instrucciones que al ser ejecutadas resuelven un problema.
Un programa está compuesto de Entrada de datos, Acciones de un algoritmo y una Salida.

Subprograma: fragmento de programa que resuelve un subproblema con entidad propia.
Permiten la reusabilidad y Distribuyen la programación.

Función: subprograma que dispone de «n» entradas, pero únicamente una salida (siempre una).

Procedimiento: subprograma que dispone de «n» entradas y «n» salidas, siendo n = {0, 2, ... n}.

Compiladores

Programa informático que traduce un programa escrito en un lenguaje de programación (fuente) a otro lenguaje de programación (objeto).

Tarea de Análisis: se trata de la comprobación de la corrección del programa fuente. Fases:
- Análisis léxico (scanner).
- Análisis sintáctico (parser).
- Análisis semántico.

Tarea de Síntesis: su objetivo es la generación de la salida expresada en el lenguaje objeto.
Fases:
- Generación de código.
- Optimización de código.

Intérpretes

Un intérprete es un programa informático capaz de analizar y ejecutar un código fuente, escritos en un lenguaje de alto nivel. Los intérpretes sólo realizan la traducción a medida que sea necesaria.

Clasificación de los lenguajes de programación

Según el nivel de ABSTRACCIÓN (según el grado de cercanía a la máquina)**:**

Lenguaje **MÁQUINA:** las instrucciones se representan en código binario directamente ejecutable por el microprocesador.

Lenguajes de **BAJO nivel**: signan a cada instrucción del lenguaje máquina un nombre nemotécnico con el fin de facilitar su uso (ensamblador).

Lenguajes de **MEDIO nivel**: algo más más cercano al humano que los anteriores.

Lenguajes de **ALTO nivel**: próximos al lenguaje natural e independientes al hardware.

Según el PROPÓSITO:

GENERAL: Aptos para todo tipo de tareas.
Lenguajes: C, C++, C#, Java, Visual Basic.

ESPECÍFICO: para un objetivo muy concreto.
Lenguajes: COBOL, SQL, ABAP, Prolog, Haskell, JS...

DE SISTEMAS: para realizar SO o drivers.
Lenguajes: C.

DE SCRIPT: para tareas varias de control y auxiliares.
Lenguajes: Shell, Bash , Dash, etc.

Según el PARADIGMA de programación (estilo de programación)**:**
Nota: pueden ser multiparadigma.

Paradigma **IMPERATIVO** (por procedimientos): programas secuenciales que divide el problema en partes más pequeñas, que serán realizadas po subprogramas.

Lenguajes: C, Pascal, BASIC, ADA, COBOL, FORTRAN.

ORIENTADOS A OBJETOS (POO): usa objetos en sus interacciones, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento.

Lenguajes: Smalltalk, C++, C#, Java, Python, Eiffel, Ruby, Scala, Swift, VB.NET, PHP, JS, etc.

FUNCIONAL: programación declarativa basado en la utilización de funciones aritméticas que no maneja datos mutables o de estado. Constituidos únicamente por funciones (puras o recursivas).
Tipos: Cálculo lambda o Lógica combinatoria.

Lenguajes: Lisp, Haskell, Miranda, Scheme, Scala, familia ML, F# (Microsoft).

LÓGICA: programación declarativa, usa predicados y y permite razonamientos deductivos o inductivos. Usado es inteligencia artificial (IA).

Lenguajes: Prolog.

DIRIGIDA A EVENTOS: un evento representa cualquier cambio en el estado del programa.
Este tipo de programación permite el uso de GUI.
Lenguajes: JavaScript, C#, VB.NET, Java, ASP.NET.

POR RESTRICCIONES (Constraints Programming)
Dedicado a búsqueda compleja de soluciones con un modelo formado por variables.
Lenguajes: basados en Prolog, Mozart.

ORIENTADA A ASPECTOS (AOP): intenta formalizar y representar de forma concisa los elementos que son transversales a todo el sistema. Eliminando la necesidad de duplicar código.
Ejemplo: el control de permisos. *checkPermisos();*
Aspects: elementos transversales modularizados. **Advices**: es una acción que hay que ejecutar.
Lenguajes: AspectJ (Java), AspectC++, Aspect (Perl), phpAspect (PHP).

Según su TRADUCCIÓN:

COMPILADOS: C.

INTERPRETADOS: JavaScript.

MIXTOS: como es el caso de Java. Primero se compila y luego se ejecuta (interpretado). Máquina Virtual.

Según la clase de TIPADO:

FUERTEMENTE tipados: no permiten conversión implícita entre tipos, deben usar de forma "segura" la conversión explícita (casting en Java).
Lenguajes: C, C++, Java, Python, Haskell, ML.

DÉBILMENTE tipados: Son más permisivos en la conversión de tipos (por ejemplo, de string a float).
Lenguajes: PHP, Javascript, Lisp, Prolog, Perl.

Según la GENERACIÓN:

1ª generación (1GL): lenguaje máquina.

2ª generación (2GL): lenguajes simbólicos.

3ª generación (3GL): lenguajes de medio y alto nivel.

4ª generación (4GL): lenguajes utilizados para propósitos específicos.

5ª generación (5GL): lenguajes de inteligencia artificial.

Representación de tipos de datos: Los tipos de datos definen el modo en que se usa el espacio (memoria) en los programas. Pueden ser predefinidos (por el sistema) o abstractos (definidos por el usuario).

Tipos de datos primitivos (Java)				
TIPO	**DESCRIPCIÓN**	**DEFAULT**	**TAMAÑO**	**RANGO**
boolean	true o false	false	1 bit	true, false
byte	entero complemento a dos	0	8 bits	-128 a 127
char	carácter unicode	\u0000	16 bits	
short	entero complemento a dos	0	16 bits	-32768 a 32767
int	entero complemento a dos	0	32 bits	-2^{31} a $2^{31}-1$
long	entero complemento a dos	0	64 bits	-2^{63} a $2^{63}-1$
float	coma flotante IEEE 754	0.0	32 bits	
double	coma flotante IEEE 754	0.0	64 bits	
Conversión de tipos de datos "casting":				
Implícita	*double < = float <= long <= int <= short <= byte*			
Explícita	*(tipo) valor_a_convertir*			
Tipado estático o dinámico				
ESTÁTICO	*int i; //típico de los lenguajes compilados*			
DINÁMICO	*i="hola" //no suele ser necesario declarar el tipo de las variables*			

Operadores aritméticos			
+	Suma	**++variable**	Pre-incremento
-	Resta	**variable++**	Post-incremento
*****	Multiplicación	**--variable**	Pre-decremento
/	División	**variable--**	Post-decremento
%	Módulo (resto división entera)		
Operadores relacionales			
<	Menor que	**>**	Mayor que
<=	Menor o igual que	**>=**	Mayor o igual que
==	Igual a	**!=**	Distinto
Operadores lógicos			
&&	AND	**!**	NOT
||	OR		
Operador ternario			
condición?si true: si false	Si la condición se evalúa como verdadera se ejecuta la instrucción después del ?, en caso contrario, la instrucción después de :		
Operadores a nivel de bits			
|	OR	**^**	XOR de bits
&	AND	**~**	NOT de bits
>>	Desplazamiento a derecha	**<<**	Desplazamiento a izquierda
Operador de asignación			
=	Asignación	**^=**	XOR y asignación
+=	Incremento y asignación	**>>=**	Desplazar asignación
-=	Decremento y asignación	**<<=**	Desplazar y asignación
%=	Módulo y asignación	**&=**	AND y asignación
***=**	Multiplicación y asignación	**|=**	OR y asignación
/=	División y asignación		
Operador escape			
\n	Nueva línea	**\'**	Comilla simple
\t	Tabulador	**\"**	Comilla doble
****	Barra invertida		

INSTRUCCIONES CONDICIONALES	
if (condición) { *// Declaraciones para ejecutar si* *// la condición es verdadera* *}*	*if (condicion1) {* *// Ejecuta cuando condicion1 es verdadero* *if (condition2) {* *// Ejecuta cuando condicion2 es verdadero* *}* *}*
if (condición) { *// Ejecuta este bloque si* *// la condición es verdadera* *} else {* *// Ejecuta este bloque si* *// la condición es falsa* *}*	*if (condición)* *declaración;* *else if (condición)* *declaración;* *…* *declaración* *else ;*
switch (expresión_a_evaluar) { *case valor1:* *declaracion1;* *break;* *case value2:* *declaracion2;* *break;* *default:* *declaracionDefault;* *}*	
BUCLES	
while (condición booleana) { *cuerpo del bucle ...* *}*	*Do {* *cuerpo del bucle ...* *}* *while (condicion);*
for (inicialización; condición; incremento) { *cuerpo del bucle ...* *}*	*for (Elemento T:Colección de obj/array) {* *cuerpo del bucle ...* *}*

VECTORES Y REGISTROS: Una array o vector es una colección de variables del mismo tipo.

Bloque 3 - Tema 4: ESTÁNDAR ANSI SQL (LENGUAJE)

SQL es el lenguaje estándar ANSI/ISO de **definición, manipulación y control** de bases de datos
relaciones. Es un lenguaje declarativo. SQL (STRUCTURES QUERY LANGUAJE, lenguaje estructurado de consultas. La última publicación de estándar es ISO/IEC 9075:2016

Esquema sentencias básicas SQL		
DDL (Definición)	**DML** (Manipulación)	**DCL** (Control)
CREATE	SELECT	COMMIT, ROLLBACK,
ALTER	INSERT	SAVEPOINT
DROP	UPDATE	GRANT, REVOKE
TRUNCATE	DELETE	

Lenguaje de definición de datos (DDL): Para las sentencias de definición utilizaremos una extensión de la Forma Normal de Backus (BNF) para especificar las cláusulas del lenguaje donde:

< > representa los símbolos no terminales del lenguaje
::= es el operador de definición
[] indica elementos opcionales
{} agrupa elementos en una fórmula
| indica una alternativa
... indica repetición

Tipo de datos (Lenguaje tipado)	
CHAR (longitud) **CHARACTER (longitud)**	Dato carácter de longitud fija
VARCHAR	Dato carácter de longitud variable
BIT (longitud)	Cadena de bits de longitud fija
BOOLEAN	Tipo de datos lógico (TRUE, FALSE)
NUMBERIC **INT** **DECIMAL**	Dato numérico decimal
INTEGER	Dato numérico entero
SMALLINT	Dato numérico entero pequeño
REAL **FLOAT (precisión)**	Dato numérico con coma flotante
DATE	Valor de fecha (YEAR, MONTH, DAY)
TIME	Valor de horas (HOUR, MINUT, SECOND)
TIMESTAMP	Valor de fecha y horas

SCHEMA: Según el estándar no se usa DATABASE se usa SCHEMA. Un esquema agrupa un conjunto de elementos de la base de datos que son propiedad de un usuario.

```
CREATE SCHEMA schema_name AUTHORIZATION owner_name
[create_table_statement];  //lista_elementos_esquema

DROP SCHEMA schema_name RESTRICT; //borra el esquema solo si
no contiene elemento alguno

DROP SCHEMA schema_name CASCADE; //borra el esquema
independientemente de si está vacío o no
```

Uso de tablas: Según la arquitectura ANSI/X3/SPARC las tablas son parte del esquema conceptual.

```
CREATE TABLE schema_name.alumnos (
 nombre VARCHAR(25),
 localidad VARCHAR(30) DEFAULT 'Toledo',
 sueldo INTEGER DEFAULT 0
);

DROP TABLE alumnos; //borra la tabla y su contenido.
DROP TABLE alumnos PURGE; //PURGE borra sin pasar a la
papelera

TRUNCATE alumnos; //borra sólo su contenido (inc. los
índices).

ALTER TABLE alumnos ADD COLUMN(
    telefono VARCHAR(20),
    email VARCHAR(100)
);

ALTER TABLE alumnos ADD fecha_nacimiento DATE NOT NULL;
ALTER TABLE alumnos MODIFY fecha_nacimiento DATE NULL;
ALTER TABLE alumnos DROP COLUMN fecha_nacimiento;
ALTER TABLE alumnos RENAME TO new_table_name;
ALTER TABLE alumnos RENAME COLUMN fecha_nacimiento TO
new_name;
```

Uso de vistas: Según ANSI/X3/SPARC utilizamos la vista para los esquemas externos o lógicos.

```
CREATE or REPLACE VIEW vpedidos AS
      SELECT proveedores.id, cantidad, precio
```

```
    FROM proveedores
    INNER JOIN pedidos ON proveedores.id = pedidos.id
    WHERE proveedores.nombre = 'Tolezon';
DROP VIEW vpedidos;
DROP VIEW vpedidos RESTRICT;
DROP VIEW vpedidos CASCADE;
```

Uso de restricciones:

Restricciones de columna:

```
CREATE TABLE alumnos(
    num_id NUMBER PRIMARY KEY,
    created_at TIMESTAMP WITH TIME ZONE NOT NULL
);
```

Restricciones: **NOT NULL, PRIMARY KEY, UNIQUE, REFERENCES, CONSTRAINT**

Restricciones de tabla:

```
CREATE TABLE alumnos (
    num_id NUMBER,
    created_at TIMESTAMP WITH TIME ZONE NOT NULL,
    CONSTRAINT pk_alumnos PRIMARY KEY(num_id)
);
```

Restricciones: **PRIMARY KEY, FOREIGN KEY, UNIQUE, REFERENCES, CONSTRAINT**

Añadir restricción en tabla (no se puede definir de esta forma es la de tipo NOT NULL):

```
ALTER TABLE alumnos ADD CONSTRAINT pk_alumnos
PRIMARY KEY (num_id);
```

CLAUSULAS EN FOREIGN KEY: existe la posibilidad de aplicar políticas especiales tras la cláusula REFERENCES al añadir una restricción de tipo FOREIGN KEY.

- **SET NULL:** Coloca nulos en todas las claves secundarias relacionadas.
- **CASCADE:** Actualiza las filas relacionadas con aquella que hemos eliminado.
- **SET DEFAULT:** Actualiza con el valor por defecto las columnas relacionadas.
- **NOTHING:** No hace nada.

```
CREATE TABLE alquileres(
  dni VARCHAR(9),
  cod_pelicula NUMBER(5),
  CONSTRAINT alquileres_pk PRIMARY KEY(dni,cod_pelicula),
  CONSTRAINT alquileres_fk1 FOREIGN KEY (dni)
    REFERENCES clientes(dni) ON DELETE SET NULL,
  CONSTRAINT alquileres_fk2 FOREIGN KEY (cod_pelicula)
    REFERENCES peliculas(cod) ON DELETE CASCADE
);
```

Restricciones a nivel global (aseriones):

Para definir una aserción utilizamos la sentencia CREATE ASSERTION:

```
CREATE ASSERTION nombre_asercion CHECK (condiciones);
```

Para borrar una aserción utilizamos la sentencia DROP ASSERTION:

```
DROP ASSERTION nombre_asercion;
```

Dominios: Según ANSI/X3

```
CREATE DOMAIN dom_ciudadades AS CHAR(2)
    CONSTRAINT ciudades_clm
        CHECK (VALUE IN ("AB", "CR", "CU", "GU", "TO"));
```

Lenguaje de manipulación de datos (DML): Formado por instrucciones capaces de añadir, cambiar o eliminar los datos de las tablas. Al conjunto de instrucciones DML que se ejecutan consecutivamente, se le llama transacción.

```
INSERT INTO proveedores (nif,nombre,provincia)
VALUES('B45000A', 'Tolezon','Madrid');

UPDATE proveedores SET provincia='Toledo'
    WHERE provincia='Madrid';
        UPDATE productos SET precio=precio*1.015;

DELETE FROM proveedores WHERE id=33;
```

Alias:

```
SELECT id_trabajo AS ID, nombre FROM trabajos;
SELECT id_trabajo "ID", nombre FROM trabajos;
```

Cálculos:

```
SELECT nombre,precio,precio*1.21 FROM articulos;
```

Condiciones:

```
SELECT tipo, modelo FROM pieza WHERE precio>3;
SELECT nombre, apellido1,apellido2 FROM alumnos WHERE
edad >= 18 AND edad <= 65;

SELECT modelo,precio FROM piezas WHERE precio BETWEEN 3
AND 8;
```

```
SELECT modelo,precio FROM piezas WHERE precio IN
(3,5,8);
```

Ordenación:

```
SELECT nombre,apellido1,apellido2 FROM alumnos ORDER BY
nombre ASC;
```

Ordenación por número de columnas:

```
SELECT nombre,apellido1,apellido2,fecha_nacimiento,id
FROM alumnos ORDER BY 2,3,1;
```

Condiciones HAVING:

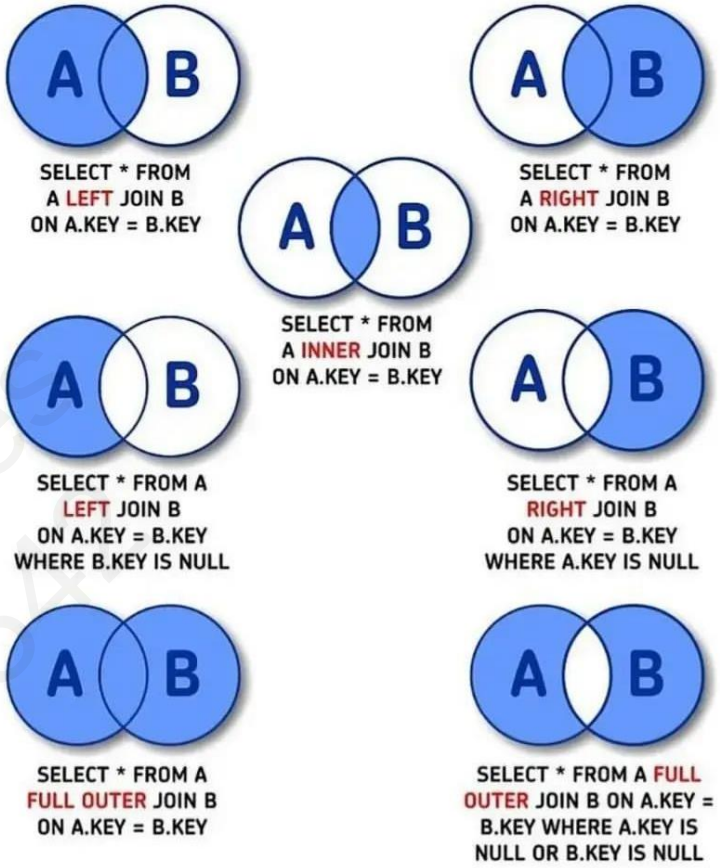
```
SELECT tipo,modelo,cantidad,SUM(cantidad)
      FROM existencias
      WHERE tipo = 2
      GROUP BY tipo,modelo
      HAVING SUM(cantidad)>500
      ORDER BY modelo;
```

Operadores			
>	Mayor que	LIKE	Búsqueda de patrones
<	Menor que	NOT LIKE	Contraposición a LIKE
>=	Mayor o igual que	IN	Pertenencia
<=	Menor o igual que	IS NULL	Valores nulos
=	Igual	IS NOT NULL	Valores no nulos
<>	Distinto	CASE	Evaluación de valores
!=	Distinto	DISTINCT	Valores distintos
AND	Conjunción de condiciones	ALL	Todos (por defecto)
OR	Disyunción de condiciones	* /	Multiplicar y dividir
NOT	Negación de condiciones	+ -	Sumar y restar
BETWEEN _ AND _	Intervalos de valores	||	Concatenación

Obtener datos de múltiples tablas:

```
SELECT nombre, apellido, departamento
      FROM departamentos,empleados
      WHERE departamentos.id_dep=empleados.id_dep
      ORDER BY nombre,apellido,departamento;
```

Clausulas (JOIN):



Lenguaje de control de datos (DCL): Se encarga fundamentalmente del control de acceso a los datos y de la integridad de estos.

La finalización de una transacción debe ser explícita con una de las siguientes sentencias:

COMMIT: todas las instrucciones de la transacción se aceptan. `COMMIT;`

ROLLBACK: todas las instrucciones de la transacción se anulan.

SAVEPOINT: Puesto que ROLLBACK anula toda la transacción, los puntos de recuperación son una utilidad que nos permite anular parte de la transacción.

```
COMMIT;
SAVEPOINT puntoA;
ROLLBACK TO SAVEPOINT puntoA;
```

Permite crear roles, permisos e integridad referencial, así como el control al acceso a la base de datos.

GRANT: otorga privilegios de acceso de usuario a la **base de datos**.

REVOKE: retira privilegios de acceso otorgados con el comando GRANT.

```
GRANT CREATE SESSION TO miusuario;
GRANT SELECT, INSERT, UPDATE, DELETE ON T_PEDIDOS TO
miusuario;
GRANT SELECT ON T_PEDIDOS TO PUBLIC;

REVOKE ALL PRIVILEGES FROM miusuario;
REVOKE ALL ON T_PEDIDOS FROM miusuario;
```

Por último, la opción **WITH GRANT OPTION** posibilita al usuario que se autoriza poder, a su vez, autorizar a otros usuarios con los mismos privilegios con los que ha sido autorizado.

Procedimientos: Oracle permite acceder y manipular información de la base de datos definiendo objetos procedurales (subprogramas) que se almacenan en la base de datos. Estos objetos procedurales son unidades de programa PL/SQL.

```
CREATE OR REPLACE PROCEDURE Actualiza_Saldo(cuenta NUMBER,
new_saldo NUMBER)
IS
  -- Declaracion de variables locales
BEGIN
  UPDATE SALDOS_CUENTAS
    SET SALDO = new_saldo,
    FX_ACTUALIZACION = SYSDATE
    WHERE CO_CUENTA = cuenta;
END Actualiza_Saldo;

ALTER PROCEDURE nombreprocedimiento …;
DROP PROCEDURE nombreprocedimiento …;

CREATE OR REPLACE FUNCTION Saldo_ok(salary REAL, title REAL)
RETURN BOOLEAN IS
      min_sal REAL;
      max_sal REAL;
BEGIN
      SELECT losal, hisal INTO min_sal, max_sal
      FROM sals
      WHERE job = title;
      RETURN (salary >= min_sal) AND (salary <= max_sal);
[EXCEPTION]
  -- Instrucciones de excepción
END;

ALTER FUNCTION nombrefuncion …;
DROP FUNCTION nombrefuncion …;

EXECUTE Actualiza_Saldo(10, 5000);
EXEC Actualiza_Saldo(10, 5000);
```

Disparadores, eventos o trigger: Un "trigger" (disparador o desencadenador) es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento (como inserción, actualización o borrado) sobre una determinada tabla (o vista); es decir, cuando se intenta modificar los datos de una tabla (o vista) asociada al disparador.

```
CREATE OR REPLACE TRIGGER TR_SALARIOS
    AFTER/BEFORE INSERT OR UPDATE OR DELETE
    ON salarios
    FOR EACH ROW
    BEGIN
INSERT INTO CONTROL
VALUES(user,SYSDATE,:OLD.id,:OLD.salario,:NEW.salario);
END TR_SALARIOS;
```

Bloque 3 - Tema 5: DISEÑO, PROGRAMACIÓN ORIENTADA A OBJETOS Y UML

DISEÑO Y PROGRAMACIÓN ORIENTADA A OBJETOS

El proceso de desarrollo orientado a objetos se compone de:
Especificación: funcionalidad de un programa y sus casos de uso.
Diseño: se plantea una solución.
Codificación o implementación: basado en los principio de Claridad y Eficiencia.
Prueba: se compila y ejecuta el programa para ver si cumple la especificación.
Mantenimiento: Correctivo para corrección de error y Evolutivo para ampliar su funcionalidad

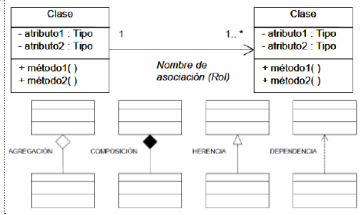
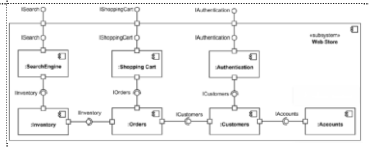
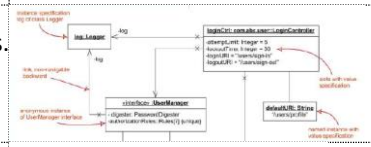
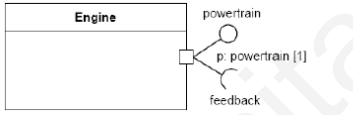
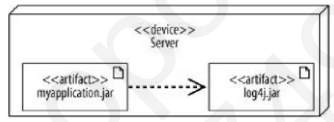
Ventajas de la orientación a objetos: Reusabilidad, menos código, interoperabilidad, extensibilidad, encapsulación, modularidad y más preparados al cambio.

PRINCIPIOS DE GENERALES DE PROGRAMACIÓN			
SOLID	Single responsibility Open/closed Liskov substitution Interface segregation Dependency inversion	Responsabilidad única Abierto/cerrado Sustitución de Liskov Segregación de interfaz Inversión de dependencia	Por Robert C. Martin para que los diseños de software sean más comprensibles, flexibles y mantenibles.
DRY	Don´t Repeat Yourself	No te repitas	Evitar duplicaciones lógicas
IoC	Inversion of Control	Delegar en terceros	Aumentar la modularidad y diseño de clases con bajo acoplamiento.
YAGNI	Your Aren´t Gona Need It	No vas a necesitar eso	No escribir código que no es necesario
KISS	Keep It Simple, Stupid	Mantelo simple, estúpido	Buscar la simplicidad y huir de la complejidad innecesaria
LoD	Law of Demeter	Ley de Demeter	Un método de un objeto solo debe interactuar con métodos del propio objeto

ELEMENTOS Y COMPONENTES SOFTWARE	
Clases:	Existen en tiempo de compilación, contienen métodos (públicos) y atributos (privados).
Objetos:	Existen en tiempo de ejecución, tienen un determinado estado (datos), comportamiento (métodos) e identidad única.
Métodos:	Definen la forma en la que los datos son manipulados (Constructor, Destructor, Getter, Settersm etc.). La interfaz estará definida por el conjunto de métodos y mensajes que soporta.
Atributos:	Son las propiedades que encapsula una clase y que definen el estado de un objeto. Público (+),Privado (-) o Protegido (#)
Mensajes:	Los objetos pueden comunicarse entre sí. El mensaje contiene nombre del objeto, el nombre de una operación y en ocasionesparámetros.
Herencia:	Una clase (subclase) adquiere las propiedades de otra clase jerárquicamente superior (superclase o clase base). Pueden ser de implementación, de interfaz, de clase, simple o múltiple.
Polimorfismo:	Es consecuencia de la herencia. El mismo método está declarado con el mismo nombre en diferentes clases. La misma operación puede comportarse de manera diferente para clases diferentes.
Sobrecarga:	Mismo nombre de métodos que reciben distintos parámetros. No confundir sobrecarga con sobreescritura (@Override).

PATRONES DE DISEÑO		
Un patrón de diseño es una solución a un problema de diseño cuya efectividad ha sido comprobada por haber sido empleada para resolver problemas similares en ocasiones anteriores.		
CREACIONALES	**ESTRUCTURALES**	**DE COMPORTAMIENTO**
Abstract Factory. Builder (Constructor). Factory Method. Prototype Singleton (Instancia única). Model View Controller (MVC).	Adapter o Wrapper. Bridge. Composite (Objeto compuesto). Decorator. Facade (Fachada). Flyweight (Peso ligero). Proxy.	Chain of Responsability. Command. Interpreter. Iterator. Mediator. Memento. Observer. State. Strategy. Template Method. Visitor.

UML (Unified Modeling Language) **La versión actual de UML es la 2.5.1 (2017)** … ISO/IEC 19505 …	
Es un lenguaje de propósito general que ayuda a especificar, visualizar y documentar modelos de sistemas software, incluido su estructura y diseño, de tal forma que se unifiquen todos sus requerimientos. No es una metodología, es un lenguaje estándar, gráfico, extensible, escalable, exhaustivo utilizando técnicas de Orientación a Objetos.	
Tipos de Diagramas:	
ESTRUCTURALES (vista estática): muestran la estructura estática del sistema a modelar. Diagrama de Clases, Componente, Objetos, Estructura Compuesta, Despliegue y de Paquetes.	De **COMPORTAMIENTO** (vista dinámica): muestran el comportamiento dinámico del sistema. Diagrama de Actividad, Casos de Uso, Máquina de Estado, Interacción, Secuencia, Comunicación Interacción y de Tiempos.

DIAGRAMAS ESTRUCTURALES (vista estática):	
Nombre y descripción:	**Notación:**
Diagrama de clases: recoge los métodos y atributos. Permite representar clases Abstractas, Interfaces y Paquetes. Relaciones: Herencia, Agregación, Composición y Dependencia.	
Diagrama de componentes: proporciona una visión física, los elementos de estos diagramas son los componentes software y las dependencias entre ellos. Elementos: Componente, Interfaz, Paquete y Relación de dependencia.	
Diagrama de objetos: Contiene los objetos con sus valores. Muestra una instantánea del sistema en un determinado momento en el tiempo.	
Diagrama de estructura compuesta: muestra la estructura interna de una clase (visión de la estructura como una caja blanca). Elementos: Parte o propiedad, Conector, Puerto y Colaboración.	
Diagrama de despliegue: muestra la disposición de las particiones físicas, se representan dos tipos de elementos, nodos y conexiones, así como la distribución de componentes del SI.	
Diagrama de paquetes: organizándolo en subsistemas, agrupando los elementos del análisis, diseño o construcción y detallando las relaciones de dependencia entre ellos. Elementos: Paquetes y Dependencias entre paquetes.	

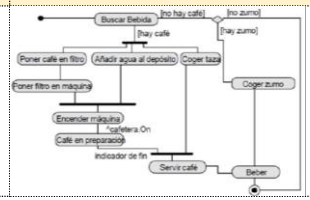
DIAGRAMAS DE COMPORTAMIENTO (vista dinámica):	
Nombre y descripción:	**Notación:**
Diagrama de actividad: muestra un flujo de actividades. Generalmente se suelen utilizar para modelar los pasos de un algoritmo. Elementos: Acción, Actividad y Transición.	

Diagrama de casos de uso: se usan para capturar los requisitos funcionales y guiar todo el proceso de desarrollo del sistema de información.
Elementos: Actores y Casos de uso

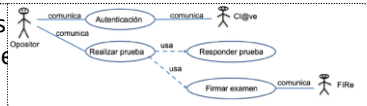


Diagrama de máquina de estado (antes diagrama transición de estados): muestra el comportamiento dependiente del tiempo de un sistema de información.
Elementos: estado, transición y pseudo-estados.

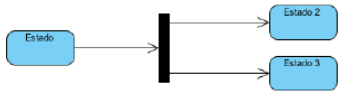


Diagrama de interacción: Incluye los diagramas de secuencia, comunicación, global de interacción y de tiempos.

Diagrama de secuencia: cuyo objetivo es describir el comportamiento dinámico del sistema de información haciendo énfasis en la secuencia de los mensajes intercambiados por los objetos.
Elementos: Objeto y línea de vida, Foco de control o activación y Mensaje.

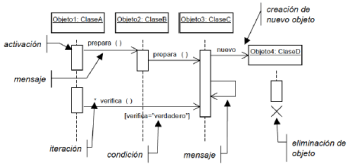


Diagrama de comunicación (antes diagrama de colaboración): cuyo objetivo es describir el comportamiento dinámico del sistema de información mostrando cómo interactúan los objetos entre sí.

Elementos: Objeto, Vínculo o enlace y Mensaje.

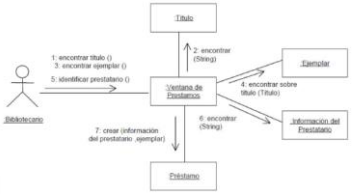


Diagrama global de interacción: representa una vista global sobre el resto de diagramas de interacción (secuencia, comunicación y tiempo).

Elementos: Interacción y Uso de Interacción.

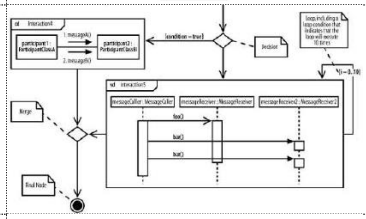
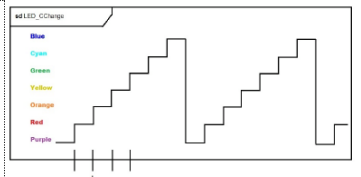


Diagrama de tiempos: se usan para mostrar el cambio en el estado o valor de uno o más elementos en el tiempo.



LENGUAJE DE PROGRAMACIÓN C (FUERTEMENTE tipado)

Lenguaje de programación estructurado creado por Brian Kernighan en los años 70.

PALABRAS RESERVADAS (estándar ANSI-C:)

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Diferencia entre mayúsculas y minúsculas. Las palabras reservadas se escriben siempre minúsculas.

Identificadores

Un identificador es cualquier palabra no reservada que comience por una letra o por un subrayado con longitud máxima 32 caracteres.
Identificadores correctos: x, _y, y22, VARIABLE, tanto_por_ciento
Identificadores erróneos: 5d, "44, por-ciento, por ciento

Tipos de datos

TIPO	TAMAÑO	MÍNIMO	MÁXIMO
unsigned char	8 bits	0	255
char	8 bits	-128	127
enum	16 bits	-32768	32767
int	16 bits	-32768	32767
unsigned int	16 bits	0	65535
short int	16 bits	-32768	32767
unsigned long	32 bits	0	4294967295
long	32 bits	-2147483648	2147483647
float	32 bits	$-3,4*10^{-38}$	$3,4*10^{-38}$
double	64 bits	$-1,7*10^{-308}$	$1,7*10^{-308}$
long double	64 u 80 bits	$-1,7*10^{-308}$	$1,7*10^{-308}$

Importante: no existe el tipo de datos STRING

Conversión de tipos de datos

AUTOMÁTICA	Convierte al tipo del operando más grande existente
EXPLÍCITA o "casting"	Forzar la conversión de un tipo de datos en otro tipo de datos. (tipo) exp.

Modificadores de acceso

MODIFICADOR	EFECTO
const	Variable de valor constante. Ejemplo: *const int x=237;*
volatile	Variable cuyo valor es modificado externamente por otro proceso. Ejemplo: *volatile const int hora;*

Variables

Globales	Accesibles desde cualquier parte del programa.
Locales	Accesibles tan solo por la función en las que se declaran.
Parámetros a la función	Accesibles de igual forma que si se declararan dentro de la función.

Importante: En C todas las variables han de ser declaradas antes de ser utilizadas.
NO PUEDE: *for(~~int cont=0~~; condición; incremento) sentencia;*
DEBE SER: *int cont; /*antes del bucle*/ for(cont=1;cont <= 3; cont++) {*

Secuencia de escape

\b	retroceso (backspace)	**\0**	nulo
\f	avance de página	****	barra invertida
\n	nueva línea	**\v**	tabulador vertical
\r	retorno de carro	**\a**	alarma
\t	tabulador horizontal	**\o**	constante octal
\"	comilla doble	\x	constante hexadecimal
\'	comilla simple		

ARRAYS, CADENAS Y PUNTEROS

Array unidimensional	*tipo nombre[tamaño];*
Array multidimensional	*tipo nombre[tam1][tam2]...[tamN];*

Además, es posible inicializar los arrays en el momento de declararlos.
Ejemplo: *float vector[3]={-3.0,5.7,-7.5};*
También es posible inicializar arrays sin ponerles el tamaño.
Ejemplo: *float vector[]={-3.0,5.7,-7.5}; char cadena[]="Esto es una cadena";*

ESTRUCTURAS Y ENUMERACIONES

Estructura	Conjunto de variables bajo el mismo nombre. *struct nombre_estructura{}vars;*
Enumeración	Conjunto de constantes enteras con nombre. *enum DIAS{lunes, martes...} dias_sem;*
typedef	Permite definir nuevos nombres a los tipos. *typedef int entero;*

PUNTEROS

Punteros	Un puntero es una variable que contiene una dirección de memoria.
operador *	devuelve la dirección de una variable de memoria.
operador &	devuelve el valor de la variable situada en la dirección que sigue.

Ejemplo: int *a,b,c;
b=15;
a=&b; /* la variable a contendrá la dirección de memoria de la variable b que es 15 */
c=*a; /* la variable c contendrá el valor 15 */

ENTRADA / SALIDA	
Lee un carácter	int getchar(void);
Escribe un carácter	int putchar(int c);
Lee un string	char *gets(char *s);
Escribe un string	int puts(const char *s);
Lee cualquier tipo	int scanf(const char *formato[,dirección,...]);

EL PREPROCESADOR	
Directiva #include	Incluir otro archivo fuente y compilarlo. Ejemplo: #include "math.h"
Directiva #define	Definir un identificador y una cadena. Ejemplo: #define UPPER 300
#if #ifdef #ifndef #else #elif #endif #include #define #undef #line #error #pragma	

Estructura de un programa en C

```
#include <stdio.h>     /* bibliotecas */

main(int argc,char *argv[],char *env[]) {   /* los parámetros son opcionales */
    int cont;
    for(cont=1;cont <= 3; cont++) { /* Inicializo bucle en la 1ª vuelta a cont=1,*/
        printf("Iteración %d\t", cont); /* incrementa en cada vuelta el cont a 1 (cont++)*/
        incrementa(); /* repito el cuerpo del bucle mientras cont<=3 */
    } /* fin del for */
     /* El %d del printf me indica que voy a representar por pantalla un número entero + \t
tabulado */
} /* fin del main() (programa principal)*/

void incrementa(){
    int a=1;
    int b=1;
    printf("a=%d b=%d\n", a,b);
    a++;b++;
} /* fin de la función incrementa */
```

LENGUAJE DE PROGRAMACIÓN C++ (FUERTEMENTE tipado)

Lenguaje de programación evolución de C con orientado a objetos creado por B. Stroustrup en 1980.

PALABRAS RESERVADAS (además de las palabras reservadas en C)

asm	bool	catch	class	delete	false	friend	inline
new	operator	private	protected	public	template	this	throw
true	try	virtual					

EXTENSIONES DE C

Comentarios simple	// Esto es un comentario simple.

Declaraciones de variables	Las declaraciones pueden ser colocadas en cualquier parte.
Visibilidad de una variable	El alcance de una variable empieza en su declaración hasta el cierre.
Operador alcance unario	Este operador (::) permite acceder al valor de la variable global.

```
int v = 7;
::v = 10.5; // Utilizar la variable global v
cout << "variable local v = " << v << "\n";
cout << "variable global v = " << ::v << "\n";  // muestra el valor 10.5
return 0;
```

ENTRADA / SALIDA	
cout	En lugar de printf usamos el flujo de salida **cout** y el operador **<<** (''colocar en'').
cin	Y en vez de scanf usamos el flujo de entrada **cin** y el operador **>>** (''obtener de'').

```
char nombre;
int num=2;
cout << "Introduzca el nombre del fichero " << num << ": ";
cin >> nombre;

cout << "El resultado es " << num << endl;   //Para generar un salto de línea utilizamos endl.
```

Tipos de datos			
bool	8 bits	false	true

C++ proporciona la capacidad de crear tipos definidos por el usuario: enum, struct, unión y class.

Ejemplo: *struct alumno { long nmat; char nombre[41]; };*

Funciones en línea (inline)

Permite sustituir, en tiempo de compilación, compilación, la llamada a una función por el código correspondiente en el punto en que se realiza la llamada mejorando el rendimiento.

Ejemplo: *inline float cubo(const float s) { return s * s * s; }* llamada: *cubo(lado)*

Parámetros por referencia

En C++ en vez de usar los punteros, en su lugar, para indicar que un parámetro de función es pasado por referencia marcamos ampersand (&) después del tipo del parámetro en la definición de función.

Ejemplo: *void permutar(int &a, int &b) // los argumentos son referencias*

Valores por defecto de parámetros en una función

Se exige que todos los argumentos con valores por defecto estén al final de la lista de argumentos.

Ejemplo: *double modulo(double x[], int n=3);*

Sobrecarga de funciones

Mismo nombre con distintos parámetros.

int max(int a, int b) *char *max(char *a, char *b)*

Calificador const			
	DECLARACIÓN	**SIGNIFICADO**	**EXPLICACIÓN**
1	**const** tipo nombre = valor;	Variable constante	No se puede modificar el valor de la variable
2	**const** tipo *nombre = valor;	Puntero variable apuntando a variable constante	No se puede modificar el valor de la variable
3	tipo ***const** nombre = valor;	Puntero constante apuntando a variable modificable	No se puede modificar la dirección, sí el valor
4	**const** tipo ***const** nombre = valor;	Puntero constante apuntando a variable constante	No se pueden modificar ni la dirección ni el valor

(1) Definir variables constantes: *const int SIZE = 5;*
(2) Especificar que un parámetro de función no es modificable: *char *strcpy(char *s1, const char *s2);*
(3) Definir un puntero constante: *char * const s2;*

Asignación dinámica de memoria		
new	*ptr = new typeName;*	Una variable creada con el operador new dentro de cualquier bloque, perdura hasta que es explícitamente borrada con delete. Puede traspasar la frontera de su bloque y ser manipulada por instrucciones de otros bloques.
delete	*delete ptr;*	

Plantillas de funciones

Permiten la creación de funciones que ejecuten las mismas operaciones sobre distintos tipos de datos.
Empiezan con la palabra reservada template.

PROGRAMACIÓN ORIENTADA A OBJETOS

Clases

```
class Punto {    // Declaración de la clase Punto.
    int _x, _y; // Coordenadas del punto.
    public: // Principio de la declaración de interface.
        void setX (const int val);
        void setY (const int val);
        int getX( ) {return _x;}
        int getY( ) {return _y;}
};
Punto apunto; // Definición del objeto apunto.
```

Asignación dinámica de memoria: new y delete

Ejemplo: Punto p = new Punto();

Constructores de una clase

El uso del constructor es tan importante que, en el caso de que el programador no defina ningún constructor para una clase, el compilador de C++ proporciona un constructor por defecto.
Ejemplo: *Punto(const int x=0, const int y=0) {_x = x; _y = y;}*
Ejemplo con inicializadores: *Punto(const int x, const int y): _x(x), _y(y) {}*
Ejemplo constructor de copia: *Punto(const Punto &p) {_x=p.getX(); _y=p.getY(); }*

Destructor

Un destructor de clase es llamado automáticamente cuando un objeto deja de ser utilizado.
Ejemplo de declaración: *~Punto();*

Puntero this

Cada objeto mantiene un puntero implícito a sí mismo denominado this puede también ser utilizado de forma explícita. Usando la palabra reservada this, cada objeto puede determinar su propia dirección.
Ejemplo: *void setX(const int val){this->_x = val; }*

Funciones o clases amigo

Una clase puede acceder a los atributos de otra clase.
Ejemplo: *friend void setX (const int val);*

Sobrecarga de operadores

```
// sobrecarga de operadores
cadena& operator= (const cadena&);
friend cadena operator+ (const cadena&, const cadena&);
friend cadena operator+ (const cadena&, const char* );
```

HERENCIA

C++ a diferencia de Java admite herencia múltiple, una clase derivada hereda de múltiples clases base.
En C++, ''hereda de'' se reemplaza por dos puntos (:).
Ejemplo: *class Puntos3D : public Punto*

Tipos de herencia

	Accesible desde su propia clase	Accesible desde una clase derivada	Accesible desde fuera de la clase
public	**SI**	**SI**	**SI**
protected	**SI**	**SI**	**NO**
private	**SI**	**NO**	**NO**

Con **protected** los miembros de clases derivadas pueden hacer referencia a miembros públicos y protegidos de la clase base sólo utilizando los nombres de los miembros.

Clases abstractas

Es una clase que no se puede instanciar. No puede existir en tiempo de ejecución.
Ejemplo: la clase Figura es una clase abstracta con una función virtual pura: area. Las clases derivadas de Figura, Cuadrado, Circulo y Elipse, redefinen esa función virtual.

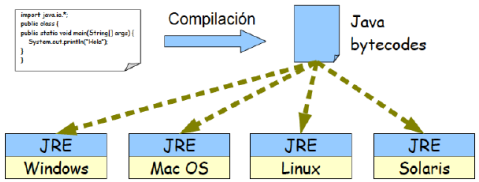
PLANTILLAS

Los tipos de datos genéricos son llamados plantillas de clase o simplemente plantillas (templates).
Las definiciones de clases plantilla empiezan con la línea: template <class T>

```
template <class T>
class Pila {    // declaración de la clase
    public:
        Pila(int nelem=10); // constructor
        void Poner(T);
        void Imprimir();
    private:
        int nelementos;
        T* cadena;
        int limite;
};
```

LENGUAJE DE PROGRAMACIÓN JAVA (FUERTEMENTE tipado)

Java es un lenguaje compilado POO independiente de la plataforma (Win, Mac, Linux, etc.)
JRE (Java Runtime Environment) generando **código binario** Java (**bytecodes**) **interpretado** por la **JVM**.



La **JDK** (Java Development Kit) incluye la JRE, el compilador de Java y las API de Java.
Como diferencia con C++, en Java no se declara un destructor de clase, dispone de un recolector de basura (**Shenandoah**) que libera de forma automática la memoria asignada a objetos sin uso.

JavaSE DK: implementación oficial de Oracle (versión actual 19 - 2022).	**GraalVM**: diseñada para acelerar el rendimiento permite mezclar lenguajes de en una sola APP
OpenJDK: es la JDK de código abierto.	**HotSpot** VM: desarrollada por Oracle (cliente/serv)

PALABRAS RESERVADAS

abstract	assert	If / else	enum	extends	final	finally	for
goto	instanceof	native	new	synchronized	throw/s	while	Etc.

NUEVAS PALABRAS RESERVADAS (soporte módulos)

open	module	requires	transitive	exports	opens	to	uses
provides	with	únicamente se consideran como tales donde aparecen módulos					

Tipos de datos

Primitivos:	representan un único valor del tamaño y formato adecuado (int, float, doublé, char, etc.).
Referencias:	representan la dirección de un objeto, no el objeto en sí (Arrays, clases e interfaces).

Tipos de datos primitivos

TIPO	TAMAÑO	DESCRIPCIÓN
byte	8 bits	Entero de un byte
boolean	8 bits	Valor booleano (true, false)
short	8 bits	Entero corto
int	16 bits	Entero
long	32 bits	Entero largo
float	32 bits	Nº en coma flotante de simple precisión
double	64 bits	Nº en coma flotante de doble precisión
char	16 bits	Un carácter Unicode
Valores por defecto:	boolean a false, char a '\0' y el resto a cero. Los tipos referencia a null.	

Caracteres especiales o de control

\b	retroceso (backspace)	****	barra invertida
\f	avance de página	**\"**	comilla doble
\n	nueva línea	**\'**	comilla simple
\r	retorno de carro	\x	constante hexadecimal
\t	tabulador horizontal		

Identificadores

Variables:	las palabras en minúscula. Si el nombre es compuesto. (Ej: unaVariable).
Constantes:	las palabras en mayúsculas separadas por el carácter subrayado (UNA_CONSTANTE).

Clases:	todo en minúscula salvo la primera letra de cada palabra (Ej: MiClase).
Clases internas:	Una clase interna es una clase que se declara dentro de otra (útiles para callbacks): *class Contenedora { … class Interna {… } } *
Herencia	Todas las clases constituyen una jerarquía de herencia con la clase java.lang.Object en la raíz. Si una clase no hereda explícitamente de otra, lo hace implícitamente de **Object**.
Interface	Únicamente puede tener firmas de métodos (sólo la declaración, sin cuerpo) y constantes *modif **interface** nombre [extends interfaz, ...] {...}*
Excepción	Una excepción es un evento que interrumpe el flujo normal de ejecución de un programa. El manejo de excepciones se encuentran en el paquete java.lang *public FileInputStream(String name) **throws** FileNotFoundException {...}*

Alcance, ámbito o scope de las variables	
Variable miembro	String miembro; *//Alcance en todo la clase*
Parámetro	Public void miFuncion (int contador){} *//Alcance en el método*
Local de bloque	int multiplicador; *//Desde la declaración hasta el final del bloque*
De exception-handler	Catch (TipoExcepcion e) {} *//Todo el bloque catch*

Estructura de un programa en Java

```
// comentario de línea
/* comentario de una o varias líneas */
/** igual que el anterior. Utilizado por javadoc para documentar app */

int[] primos = {1,2,3,5,7,9,11,13};
for (int p : primos)
System.out.println(p);

for (;;);    //compila pero no hace nada
```

NOVEDADES	
Java 8	Métodos por defecto en interfaces
Java 9	Se incorporan los módulos.
Java 10	Se incorpora la palabra reservada *var connection = "micadena";*
Java 12	Se elimina el uso de break.
Java 13	Uso triple comilla doble *String html = """*

Novedad en Java 12

```
switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> System.out.println(6);
    case TUESDAY -> System.out.println(7);
    case THURSDAY, SATURDAY, WEDNESDAY -> System.out.println(8);
}
```

Novedades en Java 13

```
String numericString = switch(integer) {
    case 0 -> {
        String value = calculateZero();
        yield value;                //yield actúa como un return
    } ;
    case 1, 3, 5, 7, 9 -> {
        String value = calculateOdd();
        yield value;
    };
    default -> {
        String value = calculateDefault();
        yield value;
    };
};
```

ARRAYS	
Array unidimensional	*String[] titulos; titulos= new String[10];*
Array multidimensional	*double[][] matriz1 = new double[3][2];*
Además, es posible inicializar los arrays en el momento de declararlos y sin ponerles el tamaño. Ejemplo: *String [] titulos= {"Nombre", "Apellidos","NIF"};*	
Su índice que varía de 0 a length-1 y el acceso a un elemento es comprobado en tiempo de ejecución Ejemplo: *for (int i = 0; i < primos.length; i++) { System.out.println(primos[i]); }*	

PAQUETES Y MÓDULOS	
Paquetes	Las clases se agrupan en paquetes. Ejemplo: *import paquete.clase;*
Biblioteca	java.io / lang / math / net / rmi / sql / util / swing / xml
java.lang	Se importa automáticamente y no es necesaria sentenciaimport

Módulos	Conjunto de clases que pueden contener uno o varios packages y las dependencias. Ventajas: Encapsulación fuerte, seguridad, optimización y desarrollo escalable. Ejemplo: *module modulo1 {* *requires java.base;* *exports app.utils;* *}*

CLASES			
[acceso] ***class MiClase*** *[extends Clase] [implements interfaces] { cuerpo de la clase }*			
public	La clase es visible en otros paquetes. Por defecto no lo es.		
abstract	La clase no puede tener instancias. Por defecto no lo es.		
final	Otras clases no pueden heredar de ésta. Por defecto no lo es.		
Variables			
Variables finales	Podemos declarar una variable final, lo que significa que no puede modificarse una vez inicializada. Para definir constantes. Ejemplo: *final double PI = 3.141592654;*		
Variables miembro de instancia u objeto	*public class User {* *private String username; }*		
Variables miembro de clase o estáticas	Sólo existe una copia para todos los objetos pensado para definir constantes. *public class User {* *static final int maxUsuarios; }*		
static	Un campo de clase	**transient**	No serializada
final	Una constante	**volatile**	Modificada por varios hilos.

Modificadores de acceso (visibilidad) para variables y métodos				
	Fuera del paquete	**Paquete**	**Clase**	**Subclase**
default	**NO**	**SI**	**SI**	**SI (si mismo paquete)**
public	**SI**	**SI**	**SI**	**SI**
protected	**NO**	**SI**	**SI**	**SI (independientemente del paquete)**
private	**NO**	**NO**	**SI**	**NO**
Referencia this				
Un objeto puede hacer referencia a sí mismo mediante la referencia especial this. Ej: *this.x = 3;*				

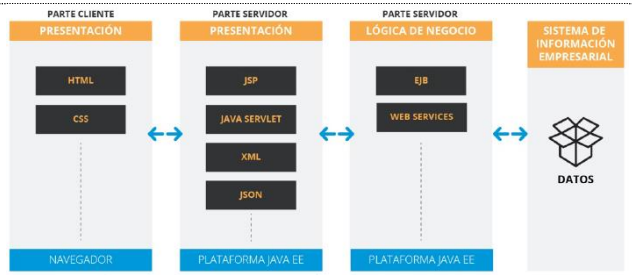
Métodos	
Métodos de instancia	Son los métodos que se aplican a un objeto.
Métodos de clase o estáticos	Estos métodos se aplican a nivel de clase. No utiliza **this** (no hay objeto)
Sobrecarga de métodos	Varios métodos con el mismo nombre en la misma clase, siempre que tipo y/o número de argumentos varíen.
Método main	Solo una tendrá un método denominado main o principal *public static void main(String[] args) {}*

Modificadores de acceso	
acceso ***[static][final][abstract][native][synchronized]*** *tipo_retorno nombre(argumentos) {...}*	
static	Método de clase en lugar de instancia. Por defecto de instancia.
final	No puede ser redefinido por las clases que extiendan ésta. Por defecto sí puede.
abstract	Se trata de un método abstracto (sin cuerpo).
native	Es un método implementado en otro lenguaje (C, C++).
synchronized	Asegura que un único **thread** ejecuta el método a la vez.

ENTRADA / SALIDA	
System.in	Entrada estándar. Ejemplo: *System.in.read()*
System.out	Salida estándar. Ejemplo: *System.out.print()*
System.err	Implementa la salida de error. Igual que System.out.

JAKARTA EE

ARQUITECTURA JAKARTA EE	
Jakarta EE es un conjunto de especificaciones y prácticas que permiten desarrollar, desplegar y gestionar aplicaciones multicapa. Consta de un conjunto de servicios, APIs y protocolos	
La plataforma Java EE 8 es la última por Oracle (2017), ahora Eclipse Foundation (Jakarta EE) versión 10. Jakarta EE modifica el nombre de los JCP a Eclipse EE.next Working Group (EE.next).	
Además de la plataforma Jakarta EE existen otras plataformas propiedad de Oracle:	
Java Card: para IoT.	**Java SE (Standard Edition)**: escritorio y servidor.
Java ME (Micro Edition): dispositivos integrados y móviles	
Arquitectura	
Capa:	Una funcionalidad de la aplicación separada del resto. Punto de vista lógico.
Nivel:	un entorno físico donde residen una o varias capas. Punto de vista físico.



(1) Nivel de cliente			
Navegador web, una aplicación independiente u otros servidores en una máquina diferente.			
(2) Nivel servidor Jakarta EE - Capa de presentación web			
Jakarta Faces (JSF):	Es una especificación de framework IU para aplicaciones web Java. Versión actual 4.0. (antes JavaServer Faces). Destaca **Eclipse MojarraTM** (incluido en Eclipse GlassFish) en un marco MVC. *import jakarta.faces.*;*		
Jakarta Servlets:	Son clases del lenguaje de programación Java. Versión actual 6.0. *import jakarta.servlet.*; (antes javax.servlet.*)* *import jakarta.servlet.http.*; (antes javax.servlet.http.*)* *public class ArgumentosPost extends HttpServlet { doPost, doGet, etc. }*		
J. Server Pages (JSP):	Las páginas JSP se transforman en servlets para ejecutarse. Versión actual 3.1. *<%@page import="java.util.*"%>* *<%! String cadena="Hola Mundo!!!"; %>* *<html><body> <%= cadena %> <jsp:include page="cuerpo.jsp"/></body></html>*		
(3) Nivel servidor Jakarta EE - Capa de negocio o lógica de negocio			
Componentes Enterprise JavaBeans	Es una clase, un componente del lado del servidor que encapsula la lógica de negocios de una aplicación. Jakarta Enterprise Beans (versión actual 4.0). *import jakarta.ejb.*; (antes javax.ejb)*		
Tipos de EJB:	Bean **de SESIÓN**: - Con estado (stateful) - Sin estado (stateless) - Singleton	Bean **dirigido POR MENSAJES** (message-driven)	Bean **de ENTIDAD** (entity): (opcional).
Servicios web RESTful JAX-RS.	WebSocket	Mail	Connectors (ERP)	Enterprise WS
	JSON Processing	WS Metadata	Persistence (JPA)	XML WS (JAX-WS)
	JSON Binding	XML RPC	Messaging (JMS)	RESTful WS (JAX-RS)
	Standard Tag Library (JSTL)	XML Registries	Transactions API (JTA)	SOAP with Attachments
	Security	JAAS (seguridad)	Authentication	Authorization
	JNDI	JDBC	StAX (XML)	JAXP (XML)
	Activation Framework (JAF)	Dependency Injection	Context Dep. Injection (CDI)	Expression Language (EL)
	Bean Validation	Managed Beans	XML Binding	NoSQL
	Concurrency	Management	Batch (XML)	

(4) Nivel servidor Jakarta EE - Capa de persistencia o acceso a datos				
JDBC:	API JDBC para acceder a los datos en un RDBMS mediante un DataSource. *import java.sql.*;*			
- DAO	Utilizaremos un Data Access Object para abstraer y encapsular todos los accesos. Clases con la conexión al sistema gestor de base de datos específico.			
ORM:	Objeto/relacional a través de frameworks que implementen JPA. *import jakarta.persistence.*;*			
- POJO	(Plain Old Java Object) es una clase simple que tiene atributos privados, constructores, métodos getters y setters.			
- Relaciones	Mediante anotaciones o un documento XML. *@Entity @Table @Id @Column(name="idxxx") @GeneratedValue @OneToOne*			
La **especificación JPA** proporciona el lenguaje JPQL y la API Criteria para consultar entidades.				
Frameworks de persistencia	Hibernate (lenguaje HQL)	EclipseLink	Apache Cayenne	jOOQ.
	Hibernate OGM (NoSQL)	Apache OpenJPA	MyBatis	Ebean
(5) Nivel de datos				
Servidores de bases de datos y otras fuentes de datos.				

Servidor Jakarta EE y contenedores	
Un **servidor Jakarta EE** es un servidor de aplicaciones que implementa las API de la plataforma Java EE y proporciona servicios estándar Jakarta EE (capa web y de negocio).	
Un **contenedor** es un proceso donde se ejecutan los componentes.	
Contenedor cliente	Se ejecuta en la máquina del cliente, enlace entre aplicación cliente y servidor.
Contenedor web	Tomcat. Protocolo AJP (Apache Jserv Protocol)
Contenedor EJB	Interfaz entre los beans empresariales que se ubica en el servidor Jakarta EE.
Los **componentes** se despliegan y ejecutan en contenedores especializados.	

Empaquetamiento y despliegue de aplicaciones	
Jakarta Archive (.jar)	basados en el formato ZIP para empaquetar un conjunto de ficheros.
Web Archive (.war)	puede ser desplegado directamente en el servidor de aplicaciones.

Enterprise Archive (.ear)	puede contener varios ficheros WAR o JAR los cuales forman los EJB.

Frameworks				
Spring	Struts 2	Play	Apache CXF	VRaptor
Spring MVC	Guice	JHipster (microserv)	Jersey	Jena
Spring Boot	PrimeFaces (JSF)	Apache Axis2	Apache Wicket	MINA

Librerías y otras herramientas			
JUnit (test funcional)	log4j	Javadoc	Ant (compilación)
DBUnit (test db)	iText (PDF)	Apache Maven	Ivy (parte Ant de Ant)
Jmeter (rendimiento)	Spring Security	Lucene (motor de busq.)	Velocity (pantillas)

.NET PLATAFORMA .NET (tipado DINÁMICO)

.NET es una plataforma de desarrollo de uso general, multiplataforma y de código abierto para crear tipos diferentes de aplicaciones. **.NET Standard** es una especificación formal de las APIs

Última versión de .NET Standard es la 2.1 y Microsoft no publicará ninguna más.

BCL (Biblioteca de Clases Base) para todas las implementaciones de .NET utiliza el espacio de nombres de System.*.

Implementaciones actuales de .NET	
.NET Framework	Entorno clásico. El runtime utilizado es CLR
.NET 6 (antes .NET Core)	De código abierto, multiOS. El runtime utilizado es coreCLR.
Mono	De código abierto (origen en Linux). El runtime es Mono runtime.
Xamarin	De código abierto, para iOS, Android y Windows con .NET
UWP (Universal Windows Platform)	Propio de Windows, usado para compilar aplicaciones Windows modernas y táctiles y software para IoT. El runtime utilizado es .NET Native para UWP.

Lenguajes de programación que ofrece .NET		
C#	F#	Visual Basic
Especificaciones del Common Language Infrastructure (CLI) estándar ECMA-335		

Common Language Runtime (CLR)
Es el entorno donde se ejecutan todas las aplicaciones .NET.
La herramienta de desarrollo compila el código fuente en un código intermedio Common Intermediate Language (**CIL**, antes MSIL o IL, Microsoft Intermediate Lenguaje). *[similar a bytecode de Java]*
Para generar dicho código el compilador se basa en el Common Language Specification (**CLS**). Para ejecutarse se necesita usa el compilador **JIT** (Just-In-Time) que genera el código máquina real.

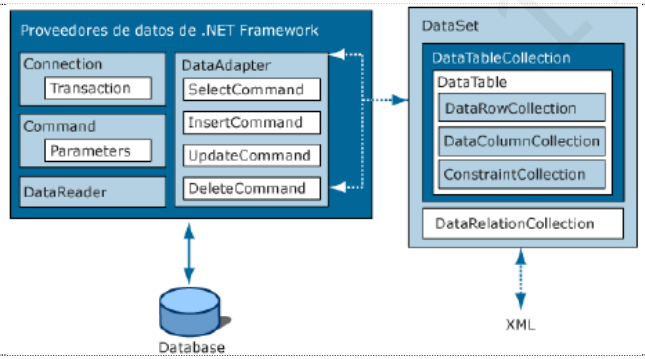
AOT (Ahead Of Time)	Es un compilador similar a JIT antes de que la aplicación se ejecute
GC, Garbage Collector	Recolector de basura de memoria automática.
CAS (Code Access Security)	CLR admite CAS como modelo de seguridad para el código administrado.
CTS, Common Type System	Sistema de tipos comunes (por ejemplo, Boolean, Byte, Char…)

Ensamblados	
.exe	Forma de un archivo ejecutable
.dll	Forma de un archivo de biblioteca
NuGet	Es un administrador de paquetes (repositorio), un paquete NuGet (*.nupkg) es un archivo ZIP que contiene los ensamblados de .NET y los metadatos asociados.
Funciones: límite de seguridad, tipos, ámbito de referencia y de versión.	
Elementos: Manifiesto, metadatos, código CIL y conjunto de recursos.	

Arquitectura	
Aplicación cliente	Windows Presentation Foundation (WPF): permite crear aplicaciones de cliente de escritorio para Windows utilizando Extensible Application Markup Language (XAML). *Similar al formato XML* *<Window xmlns="http..” Title="Windows Button" Width="250" Height="100">* *<Button Name="button">Click Me!</Button>* *</Window>*
	Windows Forms: permite crear aplicaciones de escritorio tradicionales.
	Silverlight: es un explorador, Plug-in de Multi-entorno similar a Flash.
ASP.NET (Capa de presentación)	ASP.NET Web Forms: código se ejecuta en el servidor y genera dinámicamente la salida de la página web (HTML).
	ASP.NET MVC: alternativa a Web Forms con patrón de diseño MVC *nombre**Controller*** */[Controller]/[ActionName]/[Parameters]*

	ASP.NET Web Pages: Utiliza la sintaxis Razor (no precisa experiencia previa) .cshtml: contiene el marcado HTML con código C# que usa la sintaxis Razor. .cshtml.cs: contiene código C# que controla los eventos de página. *@{var a = 4; var b = 5; } @(a*b)*
Capa de servicios	Interfaz de servicios: es como un servicio web, expone los servicios para que otro servicio web lo use. Implementados mediante WCF (Windows Communication Foundation). Los servicios web pueden ser SOAP o REST.
Capa empresarial	Fachada de la aplicación: Application Facade
	Flujos de trabajo empresariales (workflows): WF (Wind. Workflow Foundation).
	Componentes de Negocio: implementan las reglas de negocio (similar a EJB)
	Entidades de Negocio: representan los datos que se pasan entre los componentes se pueden implementar en .NET mediante ADO.NET Entity Framework. Permite utilizar clases **POCO** (Plain Old ClrObjects).
	Microsoft BizTalk Server permite conectar software diverso.

	Permite usar Language-Integrated Query (LINQ) para trabajar con distintas fuentes de datos (objetos, BD relacionales y XMLs). Parallel LINQ (PLINQ) es una variante de LINQ que permite la ejecución paralela de consultas.
Entity Framework	Es una mapeador objeto/relacional (ORM) que permite trabajar a un nivel mayor de abstracción. - Entity Framework 6 (EF6) para .NET Framework - Entity Framework Core para .NET Core
Agentes de servicios:	Permiten a los componentes empresariales acceder a un servicio externo encapsulando el acceso a un servicio.

Persistencia	
ADO.NET (ActiveX Data Object.NET)	Conjunto de clases de acceso a datos. Los dos componentes principales son: (1) Proveedores de datos (Data provider): SQL Server, OLE DB, ODBC, Oracle y EntityClient. No interactúa directamente con ningún origen de datos, usa Entity SQL. Destaca el uso de **DataReader** para solo lectura hacia delante (rendimiento). Ejemplo: *using System.Data.Odbc;* (2) DataSet: representa una caché de datos en memoria que contiene una colección de uno o más objetos DataTable. 

Frameworks				
NHibernate	Blazor (webs)	PostSharpk	MAUI (Multi-platform App UI)	
Librerías y herramientas				
Fluent NHibernate	NUnit	xUnit	log4Net	NLog
Lucene	NSwag (REST)	AutoMapper	FluentValidation	ML.NET
Managed Extensibility Framework (MEF)			ML.NET (aprendizaje automático)	
PostSharp Loggin	PostSharp MVVM	PostSharp Caching	PostSharp Threading	

LENGUAJE DE PROGRAMACIÓN C#

Lenguaje de programación multiparadigma, compilado, orientado a objetos con versionamiento. Estandarizado por ECMA-334 y ISO/IEC 23270.

PALABRAS RESERVADAS							
abstract	checked	fixed	interface	internal	lock	unchecked	unsafe
delegate	virtual	operator	override	params	private	stackalloc	Etc.

Identificadores

C# es un lenguaje sensible al contexto, diferencia entre mayúsculas y minúsculas.
Identificadores correctos: x, _y, y22, VARIABLE, tanto_por_ciento, @if, @for

Variables		
Simples	sbyte, short, int, long	Enteros con signo
	byte, ushort, uint, ulong	Enteros sin signo
	char	Caracteres Unicode (UTF-16)
	float, double	Punto flotante binario
	decimal	Punto flotante decimal alta precisión
	bool	Booleano
De enumeración	enum	Conjunto de constantes
De estructura	struct	Encapsula datos y funcionalidad

Aceptan valores NULL		Extensión de todos los demás tipos
De tupla	Ej: (double, int) t1 = (4.5, 3)	Agrupa varios elementos
De clase	Object	Clase base
	String	Cadenas
	class C {…}	Clases definidas por el usuario
De interfaz	interface I {…}	Conjunto de métodos (igual que en Java)
De matriz	Unidimensionales o multidimensionales	tipo [] tipo [,…,]
Delegado	Declaración de métodos.	delegate tipo D(…)

Tipos de datos			
TIPO	**TAMAÑO**	**MÍNIMO**	**MÁXIMO**
byte	8 bits	0	255
sbyte	8 bits	-128	127
enum	16 bits	-32768	32767
short	16 bits	-32768	32767
ushort	16 bits	0	65535
char	16 bits		
int	32 bits	-2^{31}	$2^{31}-1$
uint	32 bits	0	$2^{32}-1$
long	64 bits	-263	$2^{63}-1$
ulong	64 bits	-263	$2^{64}-1$
float	32 bits		
double	64 bits		
decimal	128 bits		
bool		false	true
La palabra clave string es un alias de String. Por lo tanto, String y string son equivalentes			
Conversión de tipos			
AUTOMÁTICA (implícita)	Conversión de tipos enteros más pequeños a más grandes		
EXPLÍCITA	(tipo)expresión		
Modificadores de acceso			
static	En la declaración se incluye el modificador static.		
const	Variable de valor constante.		
OPERADORES (se añaden)			
delegate	Crea un método anónimo que se puede convertir en un tipo delegado		
?	Operador condicional ternario. *Condición ? true : false*		
??	Operador de uso combinado NULL. *a ?? b*		
??=	Operador de asignación de uso combinado NULL. *d ??= e*		

=>	Operador lambda

Modificadores de acceso (visibilidad) para variables y métodos				
	Fuera del paquete	**Paquete**	**Clase**	**Subclase**
public	**SI**	**SI**	**SI**	**SI**
protected	**NO**	**SI**	**SI**	**SI (independientemente del paquete)**
private	**NO**	**NO**	**SI**	**NO**
internal	**Acceso desde cualquier código del mismo ensamblado**			
protected internal	**cualquier código del ensamblado en el que se ha declarado, o desde una clase derivada de otro ensamblado**			
private protected	**solo dentro de su ensamblado de declaración.**			

Estructura de un programa en C#

```
using System;
Console.WriteLine("Hello world!");

int[,] numbers2D = new int[3, 2] { { 9, 99 }, { 3, 33 }, { 5, 55 } };
foreach (int i in numbers2D) {
    System.Console.Write("{0} ", i);
}
// Output: 9 99 3 33 5 55
```

Novedad en C# 9

```
//Deja de ser necesario el denominado método estático Main y usa instrucciones de nivel superior
class Program {
    static void Main(string[] args) {
        Console.WriteLine("Hello world!");
    }
}
```

Bloque 3 - Tema 7: ARQUITECTURAS Y SERVICIOS WEB

ARQUITECTURAS DE SISTEMAS C/S Y MULTICAPA

Arquitectura de Software (vista lógica) en capas y máquinas (vista física) en niveles.

Componentes	
Capa de presentación	Interfaz de usuario.
Capa de lógica de negocio	Objetos de negocio.
Capa de datos	Almacenamiento y gestión de la información.

Características: Recursos compartidos, protocolos asimétricos, separación de funcionalidad, reutilización, independencia del hardware, integridad, escalabilidad, interoperabilidad, transparencia, encapsulación e intercambios basados en mensajes.

Arquitectura mononivel o de 1 nivel (obsoleta)

Aplicación monolítica y centralizada. Los clientes se directamente al servidor (un mainframe).

Ventajas: eficiencias y fácil instalación.

Desventajas: Mantenimiento costoso, poco escalable y baja rendimiento.

Arquitectura cliente/servidor o de 2 niveles

Aplicación en dos procesos diferentes (paradigma cliente/servidor).

Cliente (front-end)	Nivel 1	Consume servicios
Servidor (back-end)	Nivel 2	Provee servicios (Apache, Tomcat o un SGBD).

Protocolos de comunicaciones: interconectan los elementos anteriores.

Configuraciones	Presentación	Lógica de negocio	Datos
Presentación distribuida	Cliente y servidor	Servidor	Servidor
Presentación remota (cliente ligero)	Cliente	Servidor	Servidor
Lógica distribuida (JS + Servlets)	Cliente	Cliente y servidor	Servidor
Datos remotos (Monolíticas o C. Pesado)	Cliente	Cliente	Servidor
Datos distribuidos	Cliente	Cliente	Cliente y servidor

Arquitectura de 3 niveles

Almacenamiento y gestión de los datos, lógica de la aplicación y la presentación de los datos.

Cliente (front-end)	Nivel 1	Realiza las peticiones al servidor y consume servicios
Servidor web (back-end)	Nivel 2	Provee servicios (Apache, Tomcat o un SGBD).
Servidor DB (SGBD)	Nivel 3	Procesa y responde las peticiones de datos del nivel 2.

Protocolos de comunicaciones: interconectan los elementos anteriores.

Arquitectura multicapa y arquitecturas de 3 niveles y multinivel

Permite utilizar un entorno cliente ligero y evitar así el problema de la distribución.

La capa de lógica de negocio suele ser subdivida en varias capas y cumple con el artículo 7 del ENI.

Enfoque de soluciones multilaterales: arquitecturas modulares multiplataforma, reutilizar y colaborar.

Cliente (front-end)	Nivel 1	Consume servicios
Servidor web	Nivel 2	Presentación y realiza peticiones al servidor de aplicaciones.
Servidor Aplicaciones (back-end)	Nivel 3	Contiene la capa de negocio. Procesa y responde las peticiones de datos del nivel 2.
Servidor DB (SGBD)	Nivel 4	Procesa y responde las peticiones de datos del nivel 3.

Protocolos de comunicaciones: interconectan los elementos anteriores.

La capa de lógica de negocio se suele dividir en:

Capa de negocio:	Puede dividirse a su vez en distintas áreas del negocio.
Capa de seguridad:	Autenticación y autorización de usuarios y sistemas.
Capa de B. Intelligence:	Cuadro de mando estadísticas.
Capa de infraestructura:	consumo de servicios comunes.
Capa de servicios:	dedicada a la publicación de servicios
Capa de persistencia:	acceso a bases de datos y otras fuentes

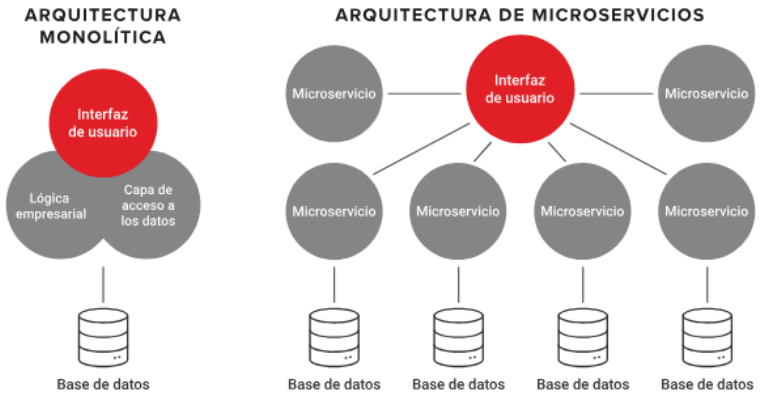
ARQUITECTURA DE MICROSERVICIOS

Consiste en construir una aplicación como un conjunto de pequeños servicios (unidad mínima) y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP).

Cada servicio es desplegado automáticamente y forma independiente.

Puede usar distintos lenguajes y tecnologías lo que facilita la gestión de equipos multifuncionales y autónomos.

La comunicación entre microservicios pueden ser gestionador por plataformas como (Openshift o Kubernetes).



ARQUITECTURA DE SERVICIOS WEB

Un servicio web es una colección de métodos independiente de la plataforma y del lenguaje.
La iniciativa ***UDDI***[4] define un servicio web como aplicaciones de negocio modulares y autocontenidas que tienen interfaces abiertos, orientados a Internet y basado en interfaces estándar. ***UDDI*** *es una iniciativa dirigida por fabricantes de software, de estándares OASIS.*
El W3C define un servicio web como un sistema software diseñado para interacción máquina-a-máquina interoperable. Normalmente descrito por WSDL que interactúa mediante mensajes SOAP XML sobre HTTP.

Tipos de servicios web	
Servicios Web SOAP	Servicios pesados mediante mensajes XML.
Servicios Web RESTful	Servicios ligeros mediante HTTP y URI.

ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

Según la **Organización para el Avance de Estándares de Información Estructurada (OASIS)** se puede definir **SOA (Arquitectura Orientada a Servicios)** como un paradigma para organizar y utilizar capacidades distribuidas.
El objetivo principal es la estrategia de construcción de servicios y no de aplicaciones finales.

CAPAS	
Sistemas operacionales	Sistemas legacy y otras aplicaciones integradas a servicios.
Componentes de servicio	Aseguran los acuerdos a nivel de servicio, y la calidad de los servicios.
Capa de servicios	Permite que estos servicios sean expuestos y descubiertos
Capa de coreografía	Establece el flujo para que los servicios actúen conjuntamente
Existirán capas transversales encargados de las tareas de seguridad, disponibilidad, etc.	
Colaboración entre servicios	
Orquestación:	Controlado totalmente por una única entidad.
Coreografía:	Colaboraciones entre cualquier tipo de aplicaciones componentes.

SERVICIOS WEB SOAP Son servicios web que usan una Arquitectura Orientada a Servicios (SOA).

Tecnologías			
Estándar	**Compañía**	**Nombre completo**	**Descripción**
XML	W3C	eXtensible Markup Language	XML se deriva de SGML (Standard Generalized Markup Language) y de HTML.
HTTP	W3C	HyperText Transfer Protocol	Protocolo de comunicación.
SOAP	W3C	Simple Object Access Protocol	Es el protocolo de comunicación basado en XML habitualmente a través de HTTP o SMTP.
WSDL	W3C	Web Service Description Language	Es el estándar propuesto para la descripción de los servicios Web.
UDDI	UDDI	Universal Description, Discovery and Integration	Directorio donde se publican los servicios Web.
- Páginas blancas		Información general sobre los proveedores de los servicios.	
- Páginas amarillas		Categorías y clasificaciones de servicios	
- Páginas verdes		Información técnica sobre los servicios.	

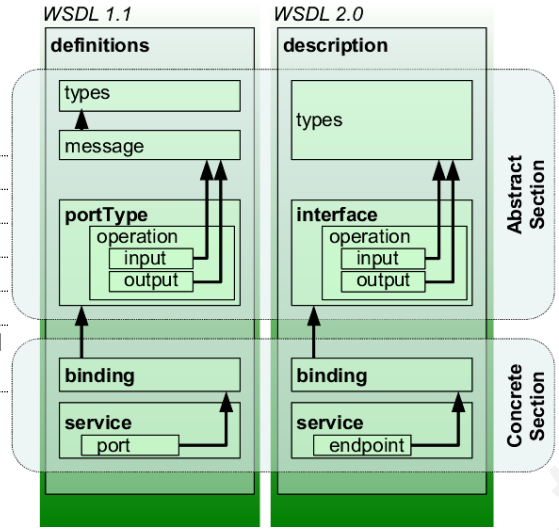
Seguridad de los servicios web del Web Service Security	
WS-Security	Extensión de SOAP de OASIS para reforzar la integridad y confidencialidad mediante tokens como SAML o Kerberos.
WS-Secure Conversation	Creado por IBM para la creación y compartición de contenidos.
WS-Trust	Estándar de OASIS que provee extensiones de WS-Security.
WS-Federation	Especificación de federación de identidades.
WS-Security Policy	Por IBM y otros, que se convirtió en un estándar de OASIS.
También disponemos de los siguientes elementos de seguridad:	
XML Digital Signature	Soporte de transacciones sin repudio y firmas del W3C.
XML Encryption	Encriptación XML del W3C que Soporta algoritmos TripleDES, AES y RSA.
XML Key Management	Registro y revocación de claves y certificados. Compuesto por la información (X-KISS) y el registro de clave (X-RKSS).
SAML:	Security Assertion Markup Language es un estándar de OASIS para autenticación y autorización.

Estructura de un mensaje SOAP: se codifica como un documento XML			
Sobre	<Envelope>	obligatorio	Describe el mensaje, a quién y cómo va dirigido. Incluye las definiciones de los tipos de datos.
Cabecera	<Header>	opcional	Para pasar información relacionada con la aplicación, los subelementos son bloques de cabecera.
	role		Especifica si un nodo concreto operará en un mensaje.
	mustUnderstand		Mara asegurar que los nodos SOAP no ignoran los bloques
	relay		Elimina el bloque de cabecera del mensaje SOAP.
Cuerpo	<Body>	obligatorio	Contiene el mensaje en sí.
Error	<Fault>	opcional	Para notificar errores al remitente de la petición.
Adjuntos	Attachment	opcional	Proporciona el contenido en el body codificado en Base64
	MTOM	opcional	Envía los datos binarios como adjunto (límite de 1MB).

WSDL (descripción)

Proporciona un modelo y un formato XML para describir servicios web entendible por la máquina.

versión 1.1	D (Definition)
versión 2.0	D (Descripción)



<description>	D (Descripción)
<types>	Tipos de datos usados
<interface>	Funcionalidad abstracta
<binding>	protocolos
<service>	conjunto de funciones
<endpoint>	dirección URI para acceder al servicio web

WSDL define cuatro tipos básicos de operaciones, **llamadas primitivas de transmisión**:
- Operación one-way (en una dirección).
- Operación request-response (petición/respuesta)
- Operación solicit-response (solicitud/respuesta)
- Operación notification (Notificación).

El W3C nos ofrece una utilidad para convertir un fichero de una versión a otra de forma automática: https://www.w3.org/2006/02/WSDLConvert.html

ARQUITECTURA REST (servicios web RESTful)

REST (Representation State Transfer) es un estilo arquitectónico que representa un modelo de comunicación HTTP mediante XML, JSON o microformatos, se conoce como servicios web ligeros.
Protocolo cliente/servidor sin estado, con una sintaxis universal (URI) y operaciones CRUD.
Los sistemas que implementan la arquitectura REST, se denominan **RESTful**.

Petición HTTP	http://www.tai.es/tema/6
Un método HTTP (HTTP Verbs)	**GET** http://www.tai.es/tema/6
XML o JSON	<alumno><nombre>Juan</nombre></alumno>
Código de estado HTTP resultado.	**200** OK HTTP/2

Acciones CRUD sobre los datos (Create, Retrieve, Update y Delete)			
POST	Crear nuevo recurso	**PUT**	Actualizar (y crear) datos
GET	Obtener datos	**DELETE**	Borrar el recurso

JWT - JSON Web Token

JWT es un es un estándar abierto (RFC 7519) mediante autenticación con token que permite firmar usando un secreto (con el algoritmo HMAC) o un par de claves pública/privada usando RSA.
JWT es un mecanismo de autenticación sin estado a que el estado del usuario nunca se almacena en el servidor, sino que se guarda en el cliente y es la aplicación la que comprobará si es válida en cada una de las peticiones. Consta de tres partes codificadas en Base64
HEADER.PAYLOAD.SIGNATURE

Códigos de estado HTTP							
Informational							
100	Continue	**101**	Switching Protocols	**102**	Processing		
Success							
200	OK	**201**	Created	**202**	Accepted	**203**	Non-authoritative
204	No Content	**205**	Reset Content				
Redirection							
300	Multiple Choices	**301**	Moved Permanently	**302**	Found	**303**	See Other
304	Not Modified	**305**	Use Proxy				
Client Error							
400	Bad Request	**401**	Unauthorized	**402**	Payment Required	**403**	Forbidden
404	Not Found	**405**	Method Not Allowed				
Server Error							
500	Internal Server Error	**501**	Not Implemented	**502**	Bad Gateway	**503**	Serv. Unavailable
504	Gateway Timeout	**505**	HTTP Version Not Supported				

FRAMEWORKS		
Apache Axis2	SOAP y RESTful	Java y C
Apache CXF	SOAP y RESTful	Java
Eclipse Jersey	RESTful	Java
WCF	SOAP y RESTful	.NET
gSOAP	SOAP	C/C++
CodeIgniter	SOAP	PHP
Zend	SOAP y RESTful	PHP
Laravel	SOAP y RESTful	PHP
OpenAPI (antes Swagger)	RESTful	**especificación**

Bloque 3 - Tema 8: APLICACIONES WEB

Una **aplicación web** es un tipo de aplicación alojada en servidor web accesible desde un navegador.

Aplicaciones NATIVAS Son desarrolla de forma específica para un Sistema Operativo.

iOS	Objective-C y Swift	Xcode
Android	Java (actualmente Kotlin) y XML	IntelliJ IDEA o Android Studio (de Google).

Aplicaciones WEB o webapp

Se ejecutan dentro del propio navegador web a través de una URL
Usan HTML, CSS y JavaScript servidas por un servidor web (como Apache o Nginx).
Un framework para desarrollo de webapp para Java es Vaadin.

Aplicaciones HÍBRIDAS

Usan HTML, CSS y JavaScript que se visualizan en un WebView con posibilidad de acceder a gran parte de las características del hardware del dispositivo.
El formato de empaquetado de la aplicación es .app (Apple Store) y .apk (Google Play).

Frameworks	
Ionic	Código abierto basado en HTML, CSS y JS puede usar Angular, React y Vue.js Plugins de Cordova o Capacitor que le da acceso al hardware del dispositivo.
Apache Cordova	Basado en HTML, CSS y JS que permite acceso al hardware del dispositivo.
Capacitor	Plugin de código abierto que permite acceso al hardware del dispositivo. También usado para aplicaciones PWA.
PhoneGap	De código abierto y propiedad de Adobe que al adquirirlo Apache le cambió el nombre de Apache Cordova en 2020.
React Native	Framework de código abierto con acceso a capacidades nativas. Utiliza JS para UI.
jQuery Mobile	Basado en HTML5 para crear sitios y apps web adaptables.
Flutter	Framework de código abierto (por Google), patrón MVC, usa lenguaje Dart también permite sistemas Windows, macOS y Linux..
Framework7	Framework de código abierto basado en HTML, CSS y JS con aspecto nativo.
Xamarin	Utiliza C# como lenguaje de programación y el IDE Visual Studio.
.NET MAUI (Multi-platform App UI):	Para APPs móviles y de escritorio nativas con C# y XAML. Es la evolución de Xamarin.Forms.
NativeScript	Utiliza JavaScript o TypeScript, no hace uso de webview, sino de un motor de renderizado propio. Por

Aplicaciones Progressive Web Apps o PWA

Son aplicaciones ejecutadas directamente en el navegador, pero con UI similar a una nativa.
Las tecnologías usadas en PWA son HTML, CSS y JavaScript, se basa en:

Service Worker	Es un fichero JS en segundo plano que se encarga de almacenar una copia de los datos en caché.
Manifiesto de la aplicación	Para especificar las características la app (manifest.json).

Frameworks			
PWABuilder	Generar aplicaciones PWA a partir de webapp existentes.		
Lighthouse	De código abierto desarrollada por Google para monitorización del rendimiento.		
Angular	**React**	**Vue.js**	**Ionic y Capacitor.**

COMPARATIVA				
	Nativas	**Web**	**Híbridas**	**Progresivas**
Coste y tiempo	Alto	Bajo	Bajo	Bajo
Rendimiento	Alto	Bajo	Bajo	Bajo
Offline	Si	No	Si	SI (cache)
Tiendas AAPs	Si	No	Si	No
Acceso hardware	Si	No	Si	No
Notificaciones Push	Si	No	Si	Si
Multiplataforma	No	Si	Si	Si
Experiencia de usuario	Alta	Baja	Baja	Progresiva

ARQUITECTURAS PARA APLICACIONES WEB	
MVC (Modelo-Vista-Controlador)	Punto de entrada a la aplicación: Controlador
MVP (Modelo-Vista-Presentador)	Punto de entrada a la aplicación: Vista
MVVM (Modelo-Vista-VistaModelo)	Punto de entrada a la aplicación: Vista
MVW (Modelo-Vista-Whatever)	Es una variante de MVC.
Flux (de Facebook)	View -> Actions ->Dispatcher -> Store(s)

DESARROLLO FRONT-END Y EN SERVIDOR, MULTIPLATAFORMA Y MULTIDISPOSITIVO		
Front-end	lado del cliente (navegador)	HTML, CSS y JavaScript Responsive Web Design (RWD)
Back-end	lado del servidor	Servlets, JSP, ASP.NET, Node.js, PHP, Ruby y Python.
Stacks disponibles (conjunto de tecnologías)		
MEAN	Mongo DB, ExpressJS, Angular y NodeJS.	
MERN	Mongo DB, ExpressJS, React y NodeJS	
MEVN	Mongo DB, ExpressJS, Vue.js y NodeJS	

HTML

HTML (HyperText Markup Language)

HTML se crea en 1980 en el CERN y es un estándar del W3C.
Las etiquetas HTML no son sensibles a mayúsculas y minúsculas.

HTML5: Versión actual del lenguaje clásico. XHTML5: variante de HTML basada en XML.

Nuevos elementos semánticos:	<nav>, <header>, <footer> y <article>
Nuevas características de formularios:	Formularios usables y accesibles.
Audio y vídeo nativos:	Incluye elementos <audio> y <video>
API de dibujo en Canvas:	<canvas> (lienzo) y la API asociada definir un área de dibujo.
Web Sockets:	Establecer una conexión continua cliente /- servidor: puerto
Aplicaciones web offline:	Las cachés y las bases de datos Web SQL.
Almacenamiento Web:	Almacenamiento basado en cookies.
Web workers:	Procesos en segundo plano para parte de los cálculos.
Geolocalización:	Define una API que permite acceder a la localización.
Elementos:	*<**p** id="parrafo1" class="pie">Texto concreto</**p**>*
Comillas:	En los elementos pueden ser simples, dobles o no usar. *<meta charset=UTF-8>*
Etiqueta única:	*<img src="logo.png">* o **
Cierre facultativa:	*<p>Parrafo sin etiqueta de cierre.* <!--Se recomienda cerrar -->
Booleanos:	*<input type="checkbox" **checked**>*
<html><head><body>	Son elementos facultativos que si no existen los añade el navegador.
Buenas prácticas:	Usar siempre las comillas, minúsculas, cerrar con etiquetas y elementos.

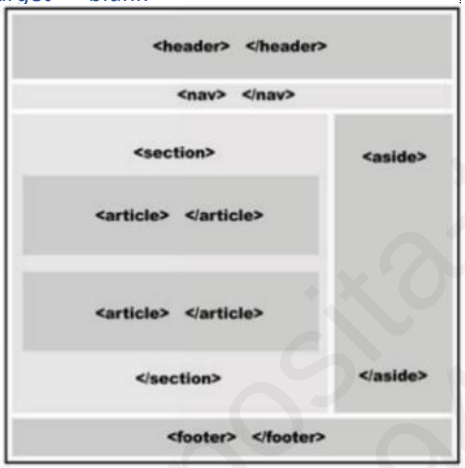
Elementos HTML

<section>	Sección en un documento.
<nav>	Sección que solo contiene enlaces de navegación.
<article>	Contenido autónomo que puede existir independientemente del documento.
<aside>	Contenido indirectamente relacionado con el resto de contenido.
<h1> al <h6>	Define encabezamientos del más importante h1 al menos h6.
<hgroup>	Encabezado de una sección.
<header>	Cabecera de la página o sección.
<footer>	Pie de página o sección.
<address>	Sección que contiene información de contacto (autor)**. *Se visualiza en cursiva.***
<main>	Define el contenido principal en el documento (sólo existe uno).
<p>	Párrafo
<hr>	Regla horizontal (elemento vacío) ----------
<pre>	Contenido preformateado
<blockquote>	<blockquote> Sección de texto que se cita de otra fuente

** **	Lista ordenada	** **	Elemento de lista
** **	Lista no ordenada		
<dl>	Lista de definiciones		
<dt>	Nombre del término		
<dd>	Descripción del término		
<figure>	Figura ilustrada como parte del documento		
<figcaption>	Leyenda de una figura		
** **	Contenedor genérico sin significado especial		
<a>	Hiperenlace	***_self / _blank***	misma / nueva pestaña
****	Texto enfatizado (cursiva)	****	Texto importante (negrita)
<small>	Texto pequeño (menor tamaño)		
<cite>	Título de una obra creativa (cursiva)	**<q cite="#">**	Cita corta (entrecomillas)
<dfn>	Definición (cursiva)	**<abbr title="">**	Abreviatura
<time datetime=>	Valor de fecha y hora		
<code>	Cógido fuente	**<var>**	Variables code (cursiva)
<samp>	Salida de un código -> code	**<kbd>**	Entrada de teclado code
<sub>	Subíndice	**<sup>**	Superíndice
<i>	Cursiva (italic)	**<b>**	Negrita (bold)
~~**<u>**~~	~~Subrayado (no en HTML5)~~	**<mark>**	Subrayado en amarillo
<ruby>	Furigana japonés	**<bdi>**	Dirección de texto (árabe)
****	Elemento en línea	** **	Salto de línea
<wbr>	Oportunidad de salto de línea	~~**<font>**~~	~~Fuente texto (no HTML5)~~
<object>	Objeto embebido (video…)	**<param>**	Parámetros de <object>
<video>	Un video <source src=>	**<audio>**	Contiene <source src=>
<source>	Permite especificar archivos	**<track>**	Especifica pistas de texto
<canvas>	Área de mapa bits	**<map>**	Mapa de imagen
<area>	Área dentro de un map		
<table>	Definición de tabla	**<caption>**	Leyenda de tabla
<colgroup>	Grupo de columnas	**<col>**	Columna/s con formato
<thead>	Cabecera de la tabla	**<tbody>**	Cuerpo de la tabla
<tfoot>	Pie de la tabla	**<tr>**	Fila
<td>	Celda (columna)	**<th>**	Celda de cabecera
<form>	Formulario	**<fieldset>**	Conjunto de elementos
<datalist>	Conjunto de valores	**<output>**	Resultado de un cálculo
<progress>	Barra de progreso	**<meter>**	Representa una medición
<keygen>	Generador de claves	**<input>**	Elementos de formulario
<details>	Representa un widget	**<summary>**	Título o leyenda <details>
<dialog>	Representa una caja de diálogo		

Estructura página HTML

```
<!DOCTYPE html>  <!-- Obligatorio -->
<html lang="es">
   <head>
        <title>Titulo de mi web</title>
        <script src="codigo.js" type="text/javascript"></script>
        <style> h1 {color:blue;}  </style>
        <link rel="stylesheet" href="estilos.css" type="text/css">
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <base href=http://www.mipagina.es/vista/ target="  blank">
   </head>
   <body>
        <section>
           <h1>Titular</h1>
           <p>Parrafo de la noticia.</p>
        </section>
        <nav class="menu">
           <ul>
              <li><a href="#">Inicio</a></li>
              <li><a href="#">Sobre nosotros</a></l
           </ul>
        </nav>
        <dl>  <!-- Lista de definiciones -->
              <dt>Cafe</dt>   <dd>Bebida caliente d
              <dt>Leche</dt>   <dd>Bebida fría de color blanco</dd>
        </dl>
        <figure>
              <img src="pic_trulli.jpg" alt="Trulli" style="width:100%">
              <figcaption> Fig.1 - Trulli, Puglia, Italy. </figcaption>
        </figure>
        <table>
              <tr>  <th>ISBN</th> <th>Titulo</th> <th>Precio</th>  </tr>
              <tr>  <td>3476896</td> <td>HTML</td> <td>40€</td>  </tr>
        </table>
   </body>
</html>
```



CSS (Cascading Style Sheets- Hojas de estilo en cascada)

Permite separar el contenido HTML y XHTML de la presentación.
Es una especificación del W3C.

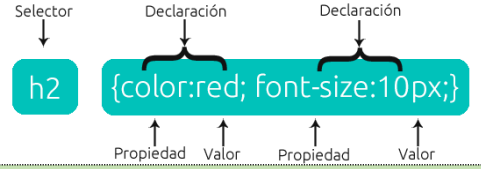
CSS3: Versión actual desde 2005. Reduce el uso de etiquetas HTML y código JavaScript.

Familias de motores de navegadores: interpretar y dibujar (render)

Familia Trident:	basados en el motor de dibujo de Internet Explorer.
Familia Gecko:	desarrollado por Mozilla.
Familia WebKit:	Usado por Safari de Apple
Familia Blink:	Es una bifurcación del proyecto WebKit. Utilizado por Opera y Google Chrome.

Incluir en el mismo HTML	*<style type="text/css"> p { color: black; } </style>*
Definir CSS en un archivo EXTERNO	*<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />*
Otra alternativa en un archivo	*<style type="text/css" media="screen and (min-width: 600px)" > @import "/css/estilos.css"; </style>*
CSS en línea (no recomendado):	* Texto. *

Estructura básica



Medios

all	Todos los medios definidos
braille	Dispositivos que emplean sistema braille
print	Impresoras y navegadores en modo Vista Previa para imprimir
screen	Pantalla
speech	Sintetizadores para navegadores de voz

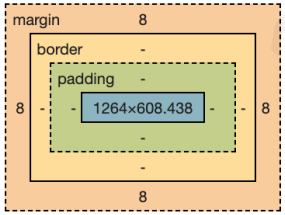
Selectores

Agrupación de selectores:	*p, a, span, em { text-decoration: underline; }*
Selector descendente:	*p a span em { text-decoration: underline; }*
Selector de clase (punto):	*.destacado { color: red; }*
Selector de ID (#):	*#destacado { color: red; }*
Combinación de selectores básicos:	*div.aviso span.especial { ... }*
Selector de hijos:	*p > span { color: blue; }*

Selector de hermanos adyacente:	*h1 + h2 { color: red } /* deben ser hermanos y aparecer juntos */*
Selector de hermanos general (siblings):	*h2~p { color:green; } /* un elemento hermano es el que existe en el mismo nivel o que tiene un padre en común */*
Selector de atributos	*a[href="http://www.ejemplo.com"] { color: blue; }*
Pseudoclases:	*:active*

:checked	*:default*	*:dir*	*:empty*	*:first-child*	*:disabled*	*:enabled*
:focus	*:hover*	*:in-range*	*:invalid*	*:last-child*	*:link*	*:not()*
:out-range	*:read-only*	*:read-write*	*:required*	*:root*	*:valid*	*:visited*

Pseudoelementos:	*::after*	*::before*	*::first-letter*
	::first-line	*::selection*	*::placeholder*

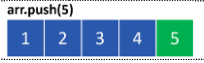
Herencia:	Una etiqueta se pasan a las etiquetas anidadas por debajo.
Colisiones de reglas:	1) Cuanto más específico sea un selector, más importancia tiene. 2) A igual especificidad, se considera la última regla indicada.
Precedencia de hojas de estilo:	Hojas de estilo de AUTOR o diseñador (propietario de la página). Hojas de estilo de USUARIO (en su perfil del navegador). Hojas de estilo de NAVEGADOR (las del navegador por defecto).
Declaración de importancia:	!important; /*prevalencia sobre otras declaraciones */
Unidades de medida:	Absolutas (pt, mm…) Relativas (em, ex…) Porcentual (%)
Colores (existen 17 definidas):	*aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow*
Valor en hexadecimal:	3 dígitos *color: #rgb* o 6 dígitos *color:#rrggbb*
Valor RGB (Red Green Blue):	valores varían entre 0 y 255 *rgb(255,0,0);*
Valor HSL (Hue Saturation Light):	*p { color: hsl(120,100%, 50%); }*
Modelo de cajas o "box model" *padding-top: 10px;* *padding-right: 5px;* *padding-bottom: 10px;* *padding-left: 5px;* *padding: 10px 5px 10px 5px;*	
Posicionamiento (position):	*static* (defecto) *relative* *absolute* *fixed*
Posicionamiento flotante (flexbox):	*.contenedor{ display:flex; }*
Fuentes:	*font-family* *font-size* *font-weight* *font-style* *@font-face*
Listas y tablas:	*list-style-type* *list-style-position* *list-style-image*

Diseño adaptativo (media queries):	*@media (max-width: 600px) and (orientation: landscape) { }*
Preprocesadores de CSS:	Compilan a CSS convencional SASS, LESS y Stylus
Bootstrap (creado por Twitter):	Framework de presentación, diseño en rejilla (12 columnas)
Otros frameworks de presentación:	Bulma, Tailwind, Semantic UI, Foundation o MiniCSS.

JAVASCRIPT (débilmente tipado)

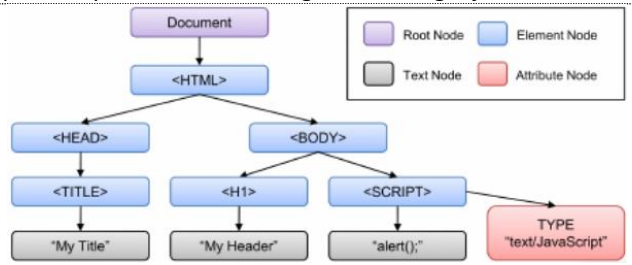
Es un lenguaje de programación interpretado, «lenguaje de scripting» del lado del cliente. Desarrollado por Netscape bajo el estándar ECMAScript 6. Comunicación asíncrona con AJAX.

Tipado dinámico:	No requiere declarar el tipo de las variables.
Interpretado, estructurad, imperativo, basado en prototipos, y sensible a mayúsculas y minúsculas	
Script INTERNO:	*<script> // Codigo.. </script>*
Script EXTERNO:	*<script src="ficheroexterno.js" type="text/javascript"></script>*
DOM:	El código no funcionará si el script es cargado antes que el código HTML
Solución (async):	*<script src="script.js" **async**></script>*
Garantizar el orden	*<script **defer** src="js/script1.js"></script>* *<script **defer** src="js/script2.js"></script>*

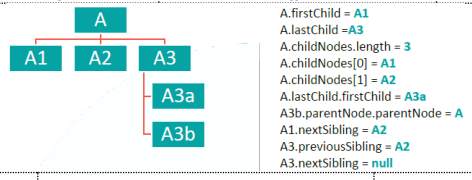
Variables (el primer carácter no puede ser un número):	***var*** (global)		***let*** (local)		***const***	
Tipos de datos:	Undefined	Boolean	Null	Number	String	Symbol	Object
Operadores:	Como otros lenguajes exepto el operador de potencia ** (2**5 = 2^5)					
Control de flujo:	*if… else*	*switch*	*try...catch*	*break*	*continue*	
	while	*do... while*	*for*	*for...in*	*for...of*	
Arrays:						
arr.indexOf("rojo"); //para buscar						
Cadenas:	*concat o (+)*	*charAt*	*length*	*indexOf*		
	split	*substring*	*toUpperCase*	*toLowerCase*		
Funciones:	*function suma(alfa, beta) { return alfa + beta; }*					
Funciones flecha:	*let cuadrado = numero => numero * numero;*					

Clases:

```
class Persona {
    constructor(nombre) {
        this.nombre = nombre;
    }
}
```

Métodos estáticos:	*static distancia(a, b) { }*
Herencia:	*extends* (al igual que en Java).
Template strings:	*var descripcion = `<strong>${nombre} ${apellidos}</strong>`;*
Transpiladores (BABEL)	Compilador que traduce un código en un lenguaje fuente a otro.
Document Object Model (DOM) Estándar del W3C Jerarquía de nodos Root > Element > Text > Attr	

document.getElementById("name");		*content = element.innerHTML;*	
.getElementsByName()	*.getElementById()*	*.getElementsByTagName()*	*.getElementsByClassName()*



A.firstChild = A1
A.lastChild =A3
A.childNodes.length = 3
A.childNodes[0] = A1
A.childNodes[1] = A2
A.lastChild.firstChild = A3a
A3b.parentNode.parentNode = A
A1.nextSibling = A2
A3.previousSibling = A2
A3.nextSibling = null

*p.firstChild.**nodeValue**="xXx";*	*document.body.**appendChild**(p);*	*p.parentNode.**removeChild**(p);*

Eventos:	onblur	onchange	onclick	onfocus	onload	onselect
	onsubmit	onmousedown	onkeypress	onkeyup	onmouseover	onmouseup

Manejadores de eventos	EN LÍNEA	*<select name="select" onchange="alert('aviso');">*
	FUNCIÓN EXTERNA	*<select name="select" onchange="miFuncion();">*
	EVENTOS	*window.onload = function() { }*

AJAX (Asynchronous JavaScript And XML)

Implementación acrónima del objeto XMLHttpRequest.
La respuesta puede ser en TEXTO o en XML.

```
<script type="text/javascript">
   function traeDatos() {
      objAJAX=new XMLHttpRequest();
      var capa=document.getElementById("salidaIP");
      capa.innerHTML="Cargando...";
      objAJAX.open("POST", "ejAJAX", true);
      objAJAX.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
      objAJAX.send();
      objAJAX.onreadystatechange=function() {
         if (objAJAX.readyState==4 && objAJAX.status==200) {
            capa.innerHTML=objAJAX.responseText;
         }
      }
   }
</script>
```

Métodos:	**abort()**	**send()**
getAllResponseHeaders()	devuelve todas las cabeceras de la respuesta del servidor.	
getResponseHeader()	devuelve una cabecera concreta de la respuesta del servidor.	
setRequestHeader()	estable una cabecera a la petición que se va a realizar.	
open(metodo,url,asincrono)	método: (**GET/POST**)	url (**https://...**) asíncrono (**TRUE/FALSE**)

Atributos:	**responseText**	**responseXML**
onreadystatechange	función encargada de manejar el estado de la petición.	
readyState	**0**: no se ha iniciado la operación	**1**: se ha abierto la comunicación
	2: se ha enviado la petición	**3**: se han comenzado a recibir datos
	4: respuesta del servidor recibida completamente.	
status	código de respuesta del servidor basado en el protocolo HTTP	
statusText	explicación que acompaña al código de respuesta HTTP del servidor.	

FRAMEWORKS				
Lado del CLIENTE	Angular	Vue.js	React	EmberJS
Lado del SERVIDOR	Next.JS	ExpressJS	Node.js	
Otros Framworks	Polymer	Meteor	Backbone.js	Closure	Redux

jQuery	La función **$**('#p1') permite acceder a elementos del DOM.
Angular (de Google)	Escrito mayoritariamente en TypeScript y es Multiplataforma
Vue.js	Desarrollo de IU para aplicaciones reactivas.
React	Desarrollo de IU usa Hooks y el patrón MVC y MVVM
React Native	Permite construir aplicaciones móviles utilizando JavaScript y React.
EmberJS	Desarrollo de IU usa el patrón MVVM
Next.JS	Framework React de desarrollo web creado sobre Node.js

Node.js	Construido con el motor JavaScript V8 de Chrome
ExpressJS	Framework web para Node.js

XML (eXtensible Markup Language)

Lenguaje de marcas (case sensitive) para describir contenido estandarizado por el W3C. La última versión es XML 1.1.

PRÓLOGO (opcional):	Declara el documento como XML y su tipo: *<?xml versión="1.0" enconding="UTF-8" ?>* *<DOCTYPE libro SYSTEM "libro.dtd">*					
CUERPO (obligatorio):	Solo puede contener un elemento raíz: *<raiz> </raíz>*					
ELEMENTOS:	Pueden tener contenido o ser elementos vacíos: *<elemento>contenido</elemento>* *<elementovacio />*					
ATRIBUTOS:	Deben ir entre comillas. *<elemento atributo="valor">*					
ENTIDADES PREDEFINIDAS (caracteres especiales):	**<**	(<)	**&**	(&)	**'**	(')
	>	(>)	**"**	(")		
Secciones CDATA:	Permiten especificar datos no procesados: *<![CDATA[* info *]]>*					
COMENTARIOS:	*<!-- Esto es un comentario -->*					

VALIDACIÓN

XML BIEN FORMADO (reglas SINTÁCTICAS)	Declaración XML.	Elemento raíz.	Elementos cerrados y anidados.
	Attr. entre comillas	Case Sensitive.	
XML VÁLIDO (reglas SEMÁNTICAS)	Bien formado y es conforme a una DTD o XML Schema.		
DTD (Document Type Definition):	En un fichero externo o en el propio XML que describe la estructura y sintaxis: *<!ELEMENT libro (titulo, editorial, autores)>* *<!ELEMENT titulo (#PCDATA)>*		
XML Schemas o XSD (XML Schema Definition):	Es un documento XML que define los elementos y atributos. Permite definir nuevos tipos de datos, herencia y namespaces		

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Libro">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Titulo" type="xsd:string"/>
```

Procesamiento de documentos XML

DOM (Document Object Model):	Objetos de un documento HTML o XML de forma jerárquica.
SAX (Simple API for XML):	Parser va procesando el XML elemento a elemento. No edita.

XSL (eXtensible Stylesheet Language)

Similar al CSS pero para XML. Lenguaje que permite definir el formato de un XML para su presentación, es un conjunto de estándares del W3C:

XSLT (XSL Transformations)	XPath (XML Path Language)	XSL-FO (XSL Formatting Objects)
XQuery (acceso a datos)	XLink (relaciones cruzadas)	XPointer (fragmentos)

Lenguajes basados en XML

RSS (Really Simple Syndication):	Formato XML para distribuir contenido en la web.
RDF (Resource Description Framework):	Intercambio de datos en web semántica. sujeto-predicado-objeto (tripleta RDF).
XBRL (eXtensible Business Reporting Language):	Información contable, financiera y tributaria.

Web semántica (Web 3.0)

Colección de tecnologías (RDF, OWL, SKOS, SPARQL, etc.) para el intercambio de datos disponibles en formato estándar, accesible y manejable. También llamada Datos vinculados.

La SEMÁNTICA:	significado de los términos.
Los METADATOS:	datos que describen otros datos.
Las ONTOLOGÍAS:	jerarquía de conceptos y sus relaciones.

Tecnologías (estandarizadas por el W3C):

RDF (Resource Description Framework)	Intercambio de datos.
OWL (Web Ontology Language)	los documentos necesita ser procesada por las aplicaciones.
SPARQL (SPARQL Protocol And RDF Query Language):	lenguaje de consulta de grafos RDF.
SKOS (Simple Knowledge Organization System):	aplicación de RDF para representar la estructura básica y el contenido. Tesauros (lista de palabras para representar conceptos)

Navegadores web

Es un programa informático que interpreta el código, HTML generalmente compuesto por interfaz de usuario, Motor de renderizado e Intérprete JavaScript.

	Motor de renderizado	Intérprete JS	Otros datos
Google Chrome	Blink.	V8.	derivado de Chromium
Microsoft Edge	Blink.	V8.	derivado de Chromium
Opera	Blink.	V8.	multiplataforma
Mozilla Firefox	Gecko.	SpiderMonkey.	multiplataforma
Safari	Webkit.	JavaScriptCore.	dispositivos Apple
Internet Explorer	Trident.	Chakra.	obsoleto
Chromium	navegador de código abierto		
Lynx	destinado a usuarios con problemas visuales y test de usabilidad		

PHP (acrónimo recursivo de PHP: **Hypertext Preprocessor**)

Es un lenguaje de código abierto del lado del servidor. Versión actual estable 8.

Scripts del lado del servidor:	Analizador PHP (módulo CGI o servidor), servidor web y navegador.
Scripts la línea de comandos:	Ejecutarlo sin necesidad de un servidor o navegador.
Aplicaciones de escritorio:	Se puede utilizar PHP-GTK para escribir aplicaciones de escritorio.
Sintaxis básica:	Etiquetas de apertura y cierre *<?php ?>* La etiqueta de cierre de un bloque implica automáticamente un (*;*)
Sentencias condicionales:	*<?php if ($expresión == true): ?>* *<?php else: ?>* *<?php endif; ?>*
Comentarios:	*// comentario de una sola línea.* *# comentario de una sola línea (estilo consola).* */* comentario multilínea */*
Variables:	Se representan $nombre (case sensitive) y tienen su ámbito dentro del contexto en el que se definen (global o local).
Variable estática:	No pierde su valor. ***static*** *$a = 0;*
Variables variables:	Toman el valor de una variable y lo tratan con el nombre *$a = 'hola';* *$$a = 'mundo';* *echo "$a $hola"; // hola mundo*

Tipos de datos:	boolean	integer	float	string	array	Boolean
	object	callable	iterable	resource	null	

Operadores:	Como otros lenguajes exepto el operador de potencia ** ($2^{**}5 = 2^5$)
Control de flujo:	*if… else* *elseif/else if* *endif;* *switch* *while* *do… while* *for* *foreach* *break* *continue*
include:	incluye y evalúa el archivo especificado. ***include*** *'fichero';*
include_once:	idéntico pero además verifica si el fichero ya ha sido incluido.
require:	en caso de fallo producirá un error fatal. ***require****('fichero');*
require_once:	si el fichero ya ha sido incluido no lo incluye de nuevo.
goto:	para saltar a otra sección del programa ***goto*** *a;* *a:*

echo	*echo "hola " . "mundo";*	**print**	*print("hola"); // print "sin paréntesis";*
strlen	obtiene la longitud de un string.	**strcmp**	compara dos de string.
str_replace	Reemplaza en un string.	**substr**	*substr($cadena, $pos_inicial[, $longitud]);*
str_shuffle	reordena aleatoriamente un string.	**strpos**	encuentra la posición de la aparición
addslashes	escapa un string con barras invertidas.	**trim, ltrim y rtrim**	

str_split	convierte un string en un array	**Strtolower ystrtoupper**	
md5	calcula el hash md5.	**sha1**	calcula el hash sha1.

Arrays

Añadir	*$arr[5] = 3;*	**Eliminar**	***unset****($arr[elemento]);*
al principio	*$array_**unshift**($arr, valor);*	**Eliminar**	***unset****($array);*
al final	*$array_**push**($arr, elemento);*	**al principio**	*$array_**shift**($arr);*
Contar	***count****($arr);*	**al final**	*$array_**pop**($arr);*
Ordenar	***sort****($arr);*	**Filtrar**	*array_**filter**($arr, function());*

Funciones

Argumentos:	Por valor (predeterminado) o por referencia.
Predeterminados:	***function*** *hacer_café($tipo = "capuchino")*
Return	***return*** es opcional, si se omite será devuelto NULL
Funciones flecha	*fn(lista_argumentos) => expresión;* Ejemplo: *$var = fn($x) => $x + $y;*

Funciones utilitarias de PHP

die	Termina el script actual.	*isset*	Comprueba si una variable existe.
empty	determina si está vacía.	*unset*	Destruye una variable especificada.
cryp	hash de cadenas (un sentido).	*mail*	Permite enviar correos.

Extensiones PHP

Biblioteca **FANN**	Fast Artificial Neural Network): Red Neuronal Artificial Rápida.		
Biblioteca **Enchant**	Diccionarios (ortografía).	Extensión **Pspell**	Ortografía (sugerencias)
Extensión **JSON**		Extensión **YAML**	
Extensión **V8js**	Motor V8 JavaScript	Extensión **Taint**	detectar vulnerabilidades
Funciones de **URL**	destacan get_headers(), parse_url(), urldecode() y urlencode().		

CLASES y acceso de variables

Públicas	*public $var;*	Privadas	*private $var2;*
Constantes	*const PI = '3,14'; //no utilizan el símbolo $.*		
Clases	*class Clase { }*		
Métodos	Pueden ser public, protected o private.		
Constructores	*function __construct() { //Pueden existir varios constructores*		
Destructores	*function __destruct() { //Solo uno por clase*		
Acceso ->	*$class = new OtherClass();* *$class->myFunc();*		
Herencia	class Espinaca **extends** Verdura { }		
Operador de Resolución de Ámbito (Paamayim Nekudotayim) **::**	Token que permite acceder a elementos estáticos, constantes, y Sobrescribir. *Foo::unMetodoEstatico();*		
self y parent	*echo **self**::$my_static . "\n";*		
Interfaces	*interface Dibujable { public function pintar($color); }*		
Clases utilitarias	Gender: Género de un nombre	Lua:	Lenguaje programación Lua.

EXCEPCIONES Y ERRORES

De forma análoga a Java hace uso de **throw** para lanzar excepciones y de **try / catch** para manejarlas.

$_GET	*$_GET['nombre']*	**$_POST**	*$_POST['nombre']*
COOKIES	***setcookie**("prueba1", "", time() - 1);*		*$dato = **$_COOKIE**['prueba'];*
SESIONES	***session_start();***	***SESION**['item'] = "ordenador";*	

Conexión a bases de datos mysqli (mysql improved)

mysqli_connect	*mysqli_close*	*mysqli_select_db*
mysqli_query	*mysqli_fetch_array*	*mysqli_fetch_row:*

FRAMEWORKS PHP

Symfony, Laravel, CodeIgniter, CakePHP, Zend, Yii y FuelPHP.

DISTRIBUCIONES:	LAMP (Linux), WAMP (Windows), MAMP (Mac) y XAMPP (cualquier S.O.) + Perl
COMPOSER:	Gestor de dependencias para PHP (composer.json)

PYTHON

Lenguaje de programación de alto nivel multiparadigma (imperativo, funcional y orientado a objetos). Código abierto. Versión 3 (última). No necesita caracteres delimitadores (como las llaves), usa una estructura de los bloques y no necesita punto y coma al final de línea.

Interpretado por **consola**	> python >>> print("Hola mundo!")			
Interpretado en **fichero**	> python hola.py			
Compilado a bytecodes	CPython (C)	Jython (Java)	IronPython (.Net)	PyPy (RPython)
Compilado a ejecutables del SO	Paquetes que compilan a un ejecutable del SO (pyinstaller).			

Comentarios:	*'''comentario multilínea'''*			
	#comentario de una sola línea (estilo consola).			
Tipos de datos:	int	float	bool	str
Operaciones:	Subcadena	c[i:j:k]	Concatena	c * n
	Copia cadena	c * n	True si c1 es subcadena de c2	c1 in c2
Funciones:	Longitud de str	len(c)	Subcadenas delim.	c.split(#)
	Carácter menor/max	min(c)/ max(c)	Mayúsculas / min.	c.upper() / c.lower()
	Convierte el primer carácter de cada palabra de c a mayúscula.			c.title()

Listas (list)

Crear	list(lista)	Posición del dato	lista.index(dato)
Longitud	len(lista)	Cuantas veces el dato	lista.count(dato)
Añadir al final	l.append(dato)	Añadir en posición	l.insert(i,dato)
Eliminar el primer	l.remove(dato)	Eliminar en posición	l.pop(indice):
Ordenar	l.sort()	Invertir el orden	l.reverse()

Tuplas (tuple):	Secuencias ordenadas de objetos de distintos tipos (inmutables).
Diccionarios (dict):	Colección de pares clave valor. {'nombre':'Juan', 'id': 1}

Variables:	Comienzan siempre por una letra o guion bajo (case sensitive).				
Estructuras de control:	*if:*	*else:*	*elif:*	*while*	*for i in range(5):*
Operadores:	Como otros lenguajes excepto **//** para división entera				
Entrada:	*input(mensaje)*	**Salida:**	*print('Hola', name, sep='')*		
Funciones:	*def bienvenida(nombre):* *print('¡Bienvenido a Python', nombre)*				
Módulos:	*import calendar* *print(calendar.month(2020, 6))*				

sys	Funciones del SO	io	Datos y ficheros	math	Matemáticas.
Os	Interfaz del SO	string	Cadenas	statistics	Estadísticas.
os.path	Rutas del sistema	datetime	Fechas y horas	random	Num. aleatorios

Clases

```
class Tarjeta:
    def __init__(self, id, cantidad = 0): # Inicializador
    self.id = id # Creación atributo id
    return
```

Librerías	Keras	Aprendizaje automático (Machine Learning)	NumPy	Cálculo matemático
	Scikit-Learn		Pandas	CSV, Excel
	PyTorch		Pillow	Imágenes.
	Matplotlib	Estáticas	SciPy	Alg. matemáticos
Frameworks	Django	Desarrollo web	Flask	Web rápidas
	Web2py	Web seguras	Pyramid	Web grandes

RUBY

Ruby

Lenguaje de programación interpretado, orientado a objetos (multiparadigma), reflexivo y Case Sensitive creado por Yukihiro Matsumoto. La versión actual es la 3.

Comentarios:	*=begin Comentario multilinea =end*					
	#: comentario de una sola línea (estilo consola).					
Tipos de datos:	*Fixnum*	*Float*	*Strings*	*Array*	*Hash*	*Range*
Variables (tipado dinámico):	Local	*_i, _mi_variable*		Global	*$counter, $COUNTER*	
	De objeto	*@nombre*		Contante	*PI=3.1416*	
	De clase	*@@signo*				

Operadores:	Como otros lenguajes excepto **//** para división entera
Estructuras de control:	if if corto unless case while for
Funciones:	*def calculate_value(x,y) return x + y end*
Clases:	*class MyClass #codigo*
Excepciones:	Para lanzar una excepción se utiliza raise.
RUBY ON RAILS (RoR)	Es un framework de desarrollo de aplicaciones. Filosofía DRY y patrón MVC. Requiere tener instalado: Ruby, SQLite3, Node.js y Yarn (gestor de paquetes).

Modelo	Active Record	**Vista**	Action View	**Controlador**	Action Controller

Bloque 3 - Tema 9: ACCESIBILIDAD, DISEÑO UNIVERSAL Y USABILIDAD.

Según el W3C la Accesibilidad Web es el acceso universal a la Web. Iniciativa **WAI** (Web Accessibility Initiative) para el acceso de las personas con discapacidad, desarrollando pautas de y herramientas de evaluación.

Pautas de accesibilidad			
Pautas de accesibilidad para las herramientas de creación de contenido (**ATAG**, Authoring Tool Accessibility Guidelines): **editores** de HTML y CMS.			
Pautas para el agente de usuario (**UAAG**, User Agent Accessibility Guidelines): **navegadores**.			
Pautas para el **contenido** web (**WCAG**): La versión en vigor es WCAG 2.1 (ISO/IEC 40500:2012), en proceso la WCAG 2.2 (2023) amplia 9 criterios y elimina el criterio 4.1.1.			
4 principios (POCR):	Perceptible, operable, comprensible y robusto.		
13 pautas:	Cada una de las cuales agrupan criterios de conformidad.		
78 criterios de conformidad o de éxito:	Se clasifican en uno de los tres niveles: A, AA y AAA.		
PRINCIPIO 1. **PERCEPTIBLE**	Pauta 1.1:	Texto alternativo	(A)
	Pauta 1.2:	Contenido multimedia dependiente del tiempo	Subtítulos (A) Audiodescripción (AA) Lengua de señas (AAA)
	Pauta 1.3:	Adaptable	Secuencia significativa (A) Orientación (AA) Identificar el Propósito (AAA)
	Pauta 1.4:	Distinguible	Uso del color (A) Control del audio (A) Contraste (mínimo) (AA) Reajuste (AA) Sonido de fondo bajo (AAA)
PRINCIPIO 2. **OPERABLE:**	Pauta 2.1:	Teclado accesible	Teclado (A) Sin trampas para el foco (A) Atajos teclas caracteres (A)
	Pauta 2.2:	Tiempo suficiente	Tiempo ajustable (A) Poner en pausa (A) Re-autentificación (AAA) Tiempos de inactividad (AAA)
	Pauta 2.3:	Convulsiones y reacciones físicas (ataques epilépticos)	Umbral de tres destellos (A) Sin destellos (AAA) Interacciones animadas (AAA)
	Pauta 2.4:	Navegación	Titulado de páginas (A) Múltiples vías (AA) Encabezados y etiquetas. (AAA) Ubicación (migas de pan) (AAA) Propósito de los enlaces (AAA)
	Pauta 2.5:	Modalidades de entrada	Gestos del puntero (A) Tamaño del objetivo (AAA)
PRINCIPIO 3. **COMPRENSIBLE:**	Pauta 3.1:	Legible	Idioma de la página (A) Abreviaturas (AAA)
	Pauta 3.2:	Previsible	Al recibir el foco (A)
	Pauta 3.3:	Asistencia a la entrada de datos	Identificación de errores (A) Etiquetas o instrucciones (A) Prevención de errores (AA)
PRINCIPIO 4. **ROBUSTO:**	Pauta 4.1:	Compatible:	Nombre, función, valor (A) Mensajes de estado (AA)
Nivel de conformidad:	**Nivel A**:	Requisitos básicos.	
	Nivel AA:	Eliminar importantes barreras de acceso.	
	Nivel AAA:	Buen nivel de accesibilidad.	
WAI-ARIA	Especificaciones técnicas de aplicaciones de Internet Enriquecidas Accesibles para las personas con discapacidad. Actualmente versión 1.2		

Herramientas de evaluación de la accesibilidad			
El W3C mantiene herramientas de evaluación de la accesibilidad web ww.w3.org/WAI/ER/tools/			
OAW	El Observatorio de Accesibilidad Web (OAW) a disposición de la Administración Pública.		
TAW	Test de Accesibilidad Web desarrollado por la Fundación CTIC		
TAW Web	Servicio online (herramientas) gratuito generando un informe HTML según WCAG 2.1		
	TAW Monitor (alertas)	TAW CMS (contenidos)	MERKUR (transformar web para móviles).
	TAW Standalone (descargable)	TAW Observatory (Grid Computing)	
eXaminator (WCAG 2.0)	Wave (WCAG 2.1 y Sec.508)	Siteimprove	
AChecker	Axe Dev Tools (ext Chrome)	Colour Contrast Analyser	
SortSite (Sección 508 EEUU)	HTML CodeSniffer	ACTF de Eclipse IDE	

Accesibilidad en aplicaciones móviles	
WCAG2ICT - Guía sobre la aplicación de WCAG 2.0 a las TI y las Comunicaciones no web.	
Propiedades del software según los criterios de conformidad de UNE-EN 301549:2022.	
Etapas:	**3º**. Elegir una muestra representativa
1º. Definir el alcance de la evaluación	**4º**. Auditar la muestra seleccionada
2º. Explorar la app	**5º**. Informar de los resultados de la evaluación
Herramientas Android	Accessibility Test Framework for Android (API)
Accessibility Scanner (herramienta de analisis)	UI Automator Viewer (Android Studio)
Herramientas iOS	
Accessibility Inspector (XCode.)	Accessibility Verifier (generación de informe)

Real Decreto 1112/2018 sobre accesibilidad de los sitios web y APPS móviles del sector público

El R.D. 1112/2018 entró en vigor el 20 de septiembre de 2018 e incorpora al ordenamiento la Directiva Europea (UE) 2016/2102.

ACCESIBILIDAD para garantizar la igualdad y la no discriminación en particular de las personas con discapacidad y de las .personas mayores.

Está basado en la nueva versión de la norma EN 301549 versión 2.1.2 "Requisitos de accesibilidad de productos y servicios TIC" que a su vez se enlaza con WCAG 2.1 (niveles A y AA).

En cada portal y aplicación móvil deberá existir una declaración de «**Accesibilidad**» que será actualizada periódicamente, como mínimo una vez al año. Deberá contener como mínimo:
a) explicación sobre aquellas partes del contenido que no sean accesibles.
b) Un enlace y descripción del mecanismo de comunicación
c) Un enlace al procedimiento de reclamación.

REVISIÓN de la accesibilidad: tanto en la fase de diseño, puesta en marcha, como periódicamente:

Sitios web	antes de 2 años	Aplicaciones móviles	antes de 3 años

Se realizarán reportes públicos a la Comisión Europea cada 3 años.

También los portales que reciben financiación pública para su mantenimiento.

Unidad responsable de accesibilidad (**URA**): para garantizar e informar del cumplimiento de los requisitos de accesibilidad. En la AGE en el ámbito de las Subsecretarías de cada Departamento.

Cada URA preparará 3 informes anuales: cumplimiento, formación, quejas y reclamaciones.

Se crea el grupo de trabajo **Red de Contactos de Accesibilidad Digital de las Administraciones Públicas.**

CONTENIDOS: perceptibles, operables, comprensibles y robustos. **POCR**

Carga desproporcionada: Con carácter excepcional y motivada podrá exceptuar el cumplimiento.

Promoción, concienciación y formación: adoptarán medidas de sensibilización y divulgación.

COMUNICACIONES: informarán sobre cualquier posible incumplimiento, habilitando una dirección de correo electrónico o un formulario, además de un teléfono u oficina física de atención.

QUEJAS O SOLICITUDES: conforme a los requisitos establecidos en la LPACAP con plazo de 20 días hábiles. La respuesta deberá incluir: la Unidad, la decisión, información accesible o plazo estimado.
En caso de silencio: desestimado.

RECLAMACIÓN: deberá dirigirse a la Unidad responsable de accesibilidad al superior jerárquico que deberá responder en plazo máximo de dos meses. Plazo de alegaciones: 10 días hábiles.

2016	Se publica la Directiva (UE) 2016/2102.
2018	Entrada en vigor RD 1112/2018.
2019	Obligación para sitios web nuevos (inc. intranets o extranets).
2020	Obligación para todos los sitios web.
2021	Obligación para las aplicaciones móviles.
2021	Los estamos presentarán su primer informe.
Informe periódico UE	Cada 3 años a la Comisión Europea.
Informes de las URA	Anualmente antes del 1 de octubre a partir del año 2020.
Declaración de accesibilidad	actualizada periódicamente, como mínimo una vez al año.
MAETD	Organismo responsable del seguimiento y presentación de informes.

Diseño universal (7 Principios)

También denominado diseño para todos, que puedan utilizar todas las personas, sin necesidad de adaptación Convención de Naciones Unidad sobre Derechos de las Personas con Discapacidad (2006).

PRINCIPIO 1	Uso equiparable.	**PRINCIPIO 2**	Uso flexible.
PRINCIPIO 3	Simple e intuitivo.	**PRINCIPIO 4**	Información perceptible.
PRINCIPIO 5	Con tolerancia al error.	**PRINCIPIO 6**	Que exija poco esfuerzo físico.
PRINCIPIO 7	Tamaño y espacio para el acceso y uso.		

Usabilidad y utilidad (usefulness)

ISO/IEC 9126	Capacidad de un software de ser comprendido aprendido, usado por el usuario.
ISO/IEC 9241	Eficiencia y satisfacción alcanzar objetivos específicos a usuarios específicos.
Principios básicos	Facilidad de aprendizaje, Flexibilidad y Robustez.
Criterios (según Jacob Nielsen)	Facilidad de aprendizaje y de memorización, Eficiencia de uso, Errores y Satisfacción.
Beneficios	Reducción de los costes, aumento de la tasa de conversión y mejora la imagen.
Ley de Hick	El tiempo que se tarda en adoptar una decisión, aumenta a medida que se incrementa el número y complejidad de las opciones.

«Diseño para todos»

Real Decreto 1494/2007, de 12 de noviembre sobre las condiciones básicas para el acceso de las personas con discapacidad a las tecnologías.

Las normas de AENOR (Asociación Española de Normalización y Certificación), el organismo de normalización reconocido oficialmente, más relevantes en el campo de la accesibilidad TIC son:

UNE 139801:2003	accesibilidad al hardware.

UNE 139802:2003	accesibilidad al software.
UNE 139803:2003	accesibilidad de contenidos web. Basada en y compatible con WCAG 2.0.

CONFIDENCIALIDAD Y DISPONIBILIDAD DE LA INFORMACIÓN EN PUESTOS DE USUARIO FINAL

Confidencialidad: información que no se revela a otros no autorizados. El modo de Acceso a la información confidencial es mediante autenticación.

identidad digital	representación única de un sujeto
Identidad Digital Europea	Para acceso a servicios públicos o privados en toda la UE.
Cartera Europea de Identidad Digital (EUDI)	Caja de herramientas común (wallet digital) interoperable de la UE, basada en normas y prácticas comunes .
Autenticación	proceso de verificar la identidad de un sujeto
Solicitante	el sujeto (usuario, proceso, aplicación o dispositivo)
Factores o esquemas de autenticación	Algo que **SE SABE** (pass o PIN) / Algo que **SE TIENE** (OTP, PUSH o TOKEN) / Algo que **SE ES** (característica biométrica)
One Time Password (OTP)	Dispositivos hardware de generación de códigos de un solo uso
Autenticador	Es un elemento que el solicitante posee (un dispositivo OTP).
Verificador	Entidad que verifica al solicitante para su autenticación.
Multi-Factor-Auth (MFA)	Proceso de autenticación que más de un factor de autenticación
Two-Factor- Auth (2FA)	Combinación de dos factores pertenecientes a distintas categorías.
Umbral de sensibilidad 	**FAR** False Acceptance Rate Tasa de falsos positivos / **FRR** False Rejection Rate Tasa de falsos negativos / **ERR** Equal Error Rate Punto FAR y FRR iguales
Rasgos biométricos	Huella dactilar / Iris / Voz / Rostro / Geometría de la mano / Firma

Disponibilidad: la información está garantizada a través de sistemas de alimentación ininterrumpida, las copias de seguridad (backups), balanceadores de carga, clusters, granjas de servidores, SAN…

SEGURIDAD EN EL DESARROLLO DE LOS SISTEMAS

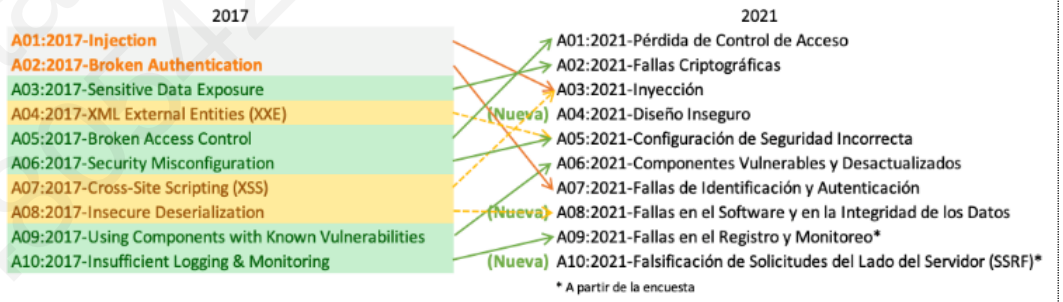
Principios de seguridad:		
Nodo Autoprotegido		Resiliencia
Análisis y Gestión del Riesgo	Defensa en Profundidad	Monitorización, Vigilancia y Respuesta a Incidentes
Mínima Funcionalidad	Control de Configuración	
Mínimo Privilegio	Verificación de la Seguridad	

Vulnerabilidades:

Desbordamiento de buffer	Inyección SQL
Condición de carrera (race condition)	Denegación del servicio
Error de formato de cadena (format string bugs)	Ventanas engañosas (Window Spoofing)
Cross-Site Scripting (XSS) ejecutar VBScript o JS	

Vulnerabilidades de seguridad en entornos web:

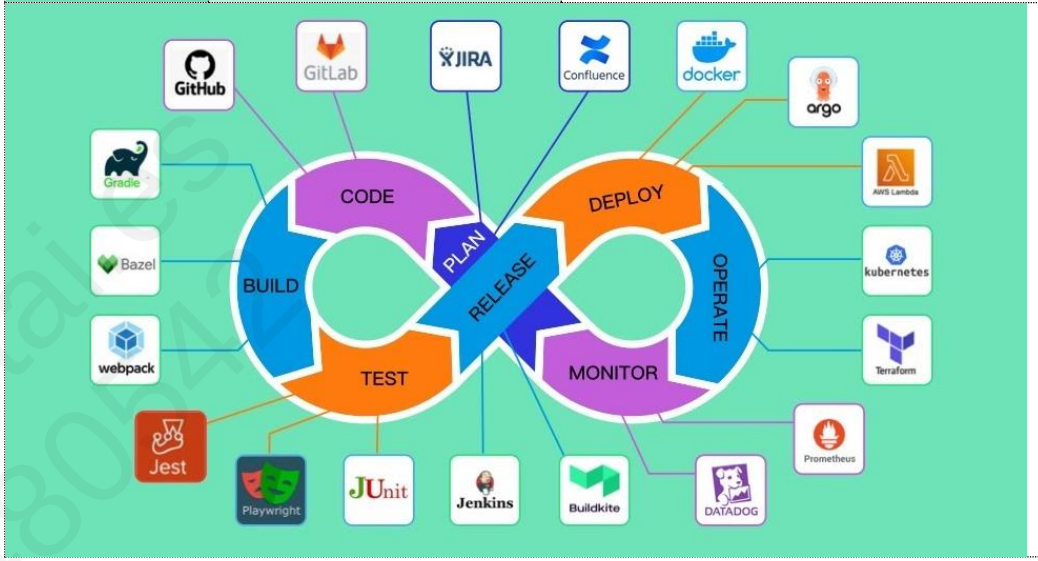
OWASP (Open Web Application Security Project): fundación sin ánimo de lucro para mejorar la seguridad del software. Publica un informe con regularidad con los 10 riesgos más importantes:



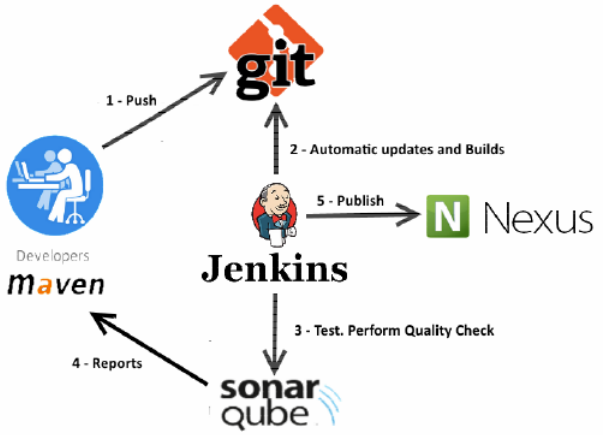
Bloque 3 - Tema 10: HERRAMIENTAS CASE, METODOLOGÍAS PRUEBAS Y CONTROL DE VERSIONES

HERRAMIENTAS CASE: CARACTERÍSTICAS	
El ciclo de vida es el conjunto de fases por las que pasa el sistema desde que nace hasta que muere. Permiten Generación de diagramas UML partir de código y viceversa, así como su análisis. El término **CASE** significa "Ingeniería del Software asistida por computador".	
Upper CASE (U-CASE)	Primeras fases (diagramas UML).
Middle CASE (M-CASE)	Fases intermedias (análisis y diseño).
Lower CASE (L-CASE)	Últimas fases (código, documentación y pruebas).
Integrated CASE (I-CASE)	Todas las fases del ciclo de vida.
CAST (Computer Aided Software Testing)	Herramientas de soporte a la prueba.
IPSE (Integrated Programming Support Environment)	Herramientas de gestión de proyectos.
MetaCASE	Permiten definir y construir herramientas CASE.

HERRAMIENTAS PARA EL DESARROLLO SOFTWARE		
Desarrollar	IDEs:	Eclipse, IntelliJ IDEA, NetBeans, Visual Studio Code, Xcode (Mac), Android Studio.
	Editores de texto:	Sublime Text, Notepad++, GNU nano, GNU Emacs.
	Control de versions	Git, GitHub, GitLab, Bitbucket, FishEye.
	Análisis (**estático**) de calidad de código:	SonarQube. Califica la calidad del código.
	Revisión de código:	Crucible, Gerrit
	Seguimiento de errores y bugs:	Jira, Redmine, Bugzilla, MantisBT, Trac.
	Herramientas para merges:	KDiff3, DiffMerge, P4Merge, JDiff.
Build	Integración continua (CI):	Jenkins (open source y escrito en Java), Bamboo, CircleCI, CruiseControl, Go.CD, Travis CI.
	Automatización:	Apache Maven, Apache Maven, Make, Gradle, Kotlin DSL, MSBuild (.NET).
Test	Continuo web:	Selenium.
	Continuo móvil:	Appium.
	Iterativo:	Lux, Expect.
	Para criterios de aceptación:	FitNesse, FIT, EasyAccept.
	Unitarios:	XUnit, JUnit, Unit.js, ATF.
	Servicios web:	SoapUI y Postman (RESTful).
Empaquetado	Repositorio de artefactos:	Nexus 3, Artifactory.
	Contenedores:	Docker.
	Gestión documental:	Confluence (wiki), Owncloud.
Gestión de la configuración		Puppet, Salt, Chef, Ansible, Vagrant, Terraform y GitOps
Despliegue	Orquestación de infraestructura:	Gradiant ITBox, MaaS, Openstack, Kubernetes.
	Orquestación de servicios:	Docker Compose, Docker Swarm.
	Orquestación multicloud:	Gradiant ITBox, Cloudify, Apache Brooklyn.
	Virtualización:	Docker, KVM (Kernel-based Virtual Machine).
	Cloud públicos:	Amazon WS, Microsoft Azure, Google CE.
Monitorización	Rendimiento:	Zabbix, Nagios, Influxdb, Telegraf, Grafana.
	Revisión de logs:	GrayLog, Logstash, Kibana, Elasticsearch.
	Experiencia y analíticas:	Piwik PRO, Matomo (PHP/MySQL), Google Analytics.



Sistema de integración continua (CI/CD):



CI/CD es la sigla para la integración y la distribución o implementación continuas, cuyo objetivo es mejorar y agilizar el ciclo de vida de desarrollo del software.

Integración continua (CI)	Incorporar los cambios de código a un repositorio compartido de código fuente de forma automática y periódica.
Distribución o implementación continua (CD)	Es un proceso en el que se integran, prueban y distribuyen los cambios de código.

GENERACIÓN DE CÓDIGO Y DOCUMENTACIÓN

Un generador de código fuente es una herramienta que a partir de diagramas UML (clases, secuencia) genera de manera automática el código.

Altova Umodel:	Genera código Java, C++, C# o VB.NET
Visual Paradigm:	Genera código Java, C++, C#, VB.NET, Python, PHP o Ruby

Generación automática de documentación

Javadoc	Java	phpDocumentor	PHP	DocFX	C#, VB y F#
JSDoc	JavaScript	NaturalDocs	C#	SandCastle	.NET
Doxygen	C++	Sphinx	Python	VSdocman	C# y VB.NET
YARD	Ruby				

MANTENIMIENTO DEL SOFTWARE

METRICA v3 distingue entre 4 tipos de mantenimiento.

CORRECTIVO:	corregir errores.	EVOLUTIVO:	incorporaciones,
ADAPTATIVO:	afectan a los entornos	PERFECTIVO:	mejorar la calidad interna

Es posible medir la complejidad que representa el mantenimiento basado en medir el desorden (**entropía del software**).

Ingeniería inversa y reingeniería

Ingeniería Inversa (Reverse Engineering):	El proceso de analizar un sistema para identificar los componentes y las interrelaciones entre ellos
Ingeniería hacia delante o directa (Forward Engineering):	El proceso que va desde un alto nivel de abstracción, hasta la propia implementación física del sistema.
Reingeniería (Reengineering):	Modificación de un sistema para ser reconstruido de una forma nueva
Reestructuración (Reestructuring):	Transformación de una forma de representación del sistema en otra distinta

Problemas del código fuente

Código espagueti:	Complejas estructuras de control de flujo. (if else anidados).
Código ravioli:	Clases simples que al interactuar entre sí se vuelven muy complejas.
Código lasaña:	Muchas capas que ante el cambio en una capa conlleva cambios en otras capas.
Código pizza:	Aquel que tiene una arquitectura plana.
Overengineering (sobreingeniería):	Código más complicado de lo que debería.
Scattered code (código disperso):	Definición de un comportamiento dónde las líneas de código están distribuidas en diferentes lugares de la aplicación.
Tangled code (código enmarañado):	Un mismo módulo implementa diferentes aspectos del sistema de forma simultánea.
"Code smell" (código que huele):	Código mal diseñado candidato a ser reestructurado.

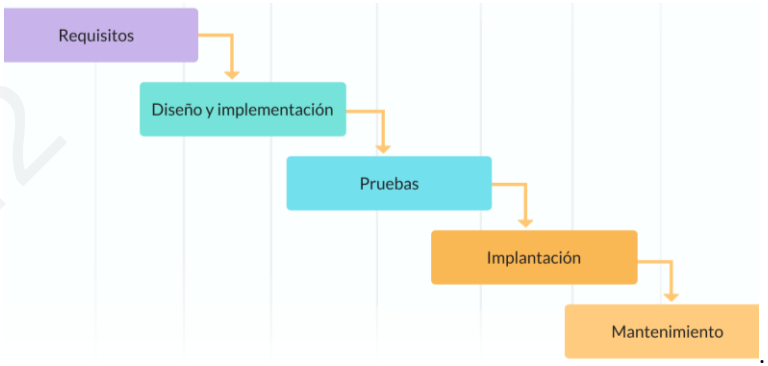
METODOLOGÍAS DE DESARROLLO SOFTWARE

Una metodología es un conjunto integrado de técnicas durante el ciclo de vida de un proyecto.

Metodologías tradicionales (formales o pesadas): fuerte planificación.

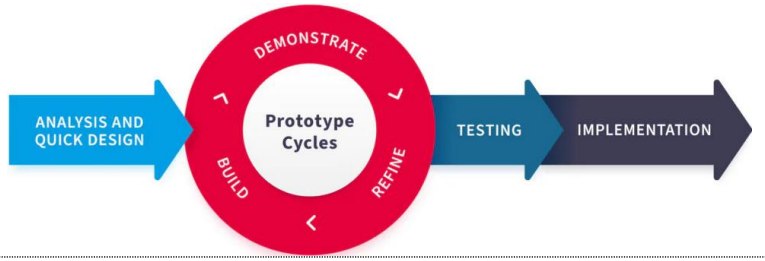
Waterfall o Cascada

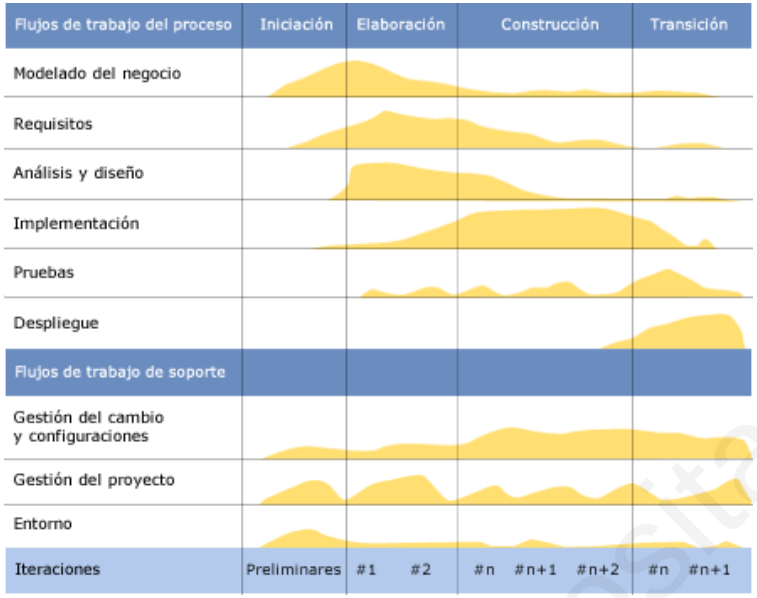
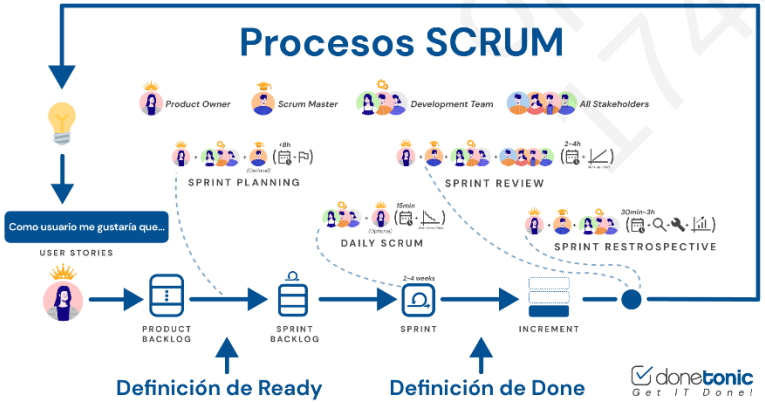
Ejecutar de forma secuencial cada avance evitando pasar a la siguiente si la misma no está concluida satisfactoriamente.

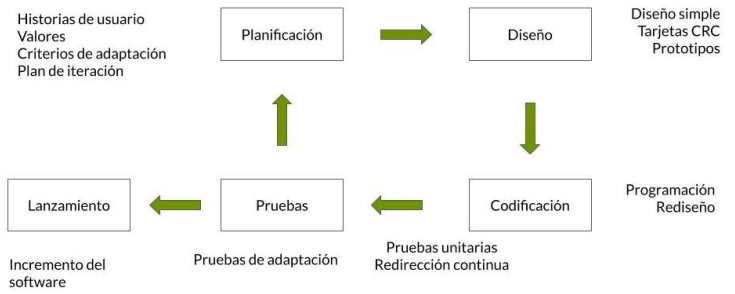


Diseño Rápido de Aplicaciones (RAD)

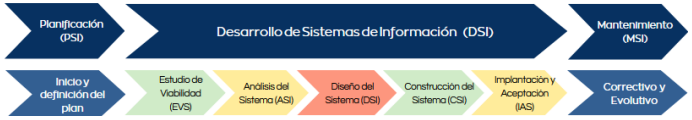
Se elabora un prototipo para que los usuarios lo prueben y poder identificar de forma directa las necesidades y requerimientos.
Se crean prioridades según la velocidad de ejecución de las actividades.
Fases: planificación de requisitos > diseño de usuario > construcción rápida > transición.



Proceso Racional Unificado (RUP)	Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. 
Metodologías ágiles: respuesta al cambio	
SCRUM	

Extreme Programming o XP	Centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Se debe definir previamente las cuatro variables que posee el proyecto que son Coste, Tiempo, Calidad y Alcance.  El cliente define el valor de negocio a implementar > el programador estima el esfuerzo > construye ese valor > vuelve al paso 1. Roles: programados, clientes, tester, tracker, coach, consultor y geestor (big boss).
Kanban (Toyota)	Visualizar lo que hace > Limitar la cantidad de trabajo en proceso > Seguimiento del tiempo > Lectura fácil de indicadores visuales > Identificar los cuellos de botella y descartar no desables. 

MÉTRICA versión 3 es Metodología de Planificación, Desarrollo y Mantenimiento de SI.
Paradigmas de desarrollo (Estructurado y Orientado a objetos).
El enfoque principal consiste en descomponer cada uno de los procesos en actividades, y éstas a su vez en tareas. PROCESOS > ACTIVIDADES > TAREAS



Proporciona también cuatro **interfaces** des orientadas a la mejora de los procesos principales:
Gestión de Proyectos, Seguridad MAGERIT, Aseguramiento de Calidad y Gestión de la Configuración.

Perfiles participantes en el desarrollo:		
Directivo – Comité de Dirección – Comité de Seguimiento – Directores de usuarios – Usuarios expertos	**Jefe de proyecto** – Responsable de Implantación – Resp. de Mantenimiento – Resp. de Operación – Resp. de Sistemas – Resp. de Seguridad – Resp. de Calidad	**Consultor** – Consultor Informático – Consultor de las TI – Consultor de SI – Especialista en Comunicaciones – Técnico de Sistemas – Técnicos de Comunicaciones
Analista – Administrador de BBDD – Equipo de Arquitectura – Equipo de Formación – Equipo de Implantación – Equipo de Operación – Equipo de Seguridad – Equipo de Soporte Técnico – Equipo de Proyecto – Grupo de Aseg. de la Calidad	El perfil de **Programador** hace referencia únicamente al participante Programador descrito en MÉTRICA Versión 3	

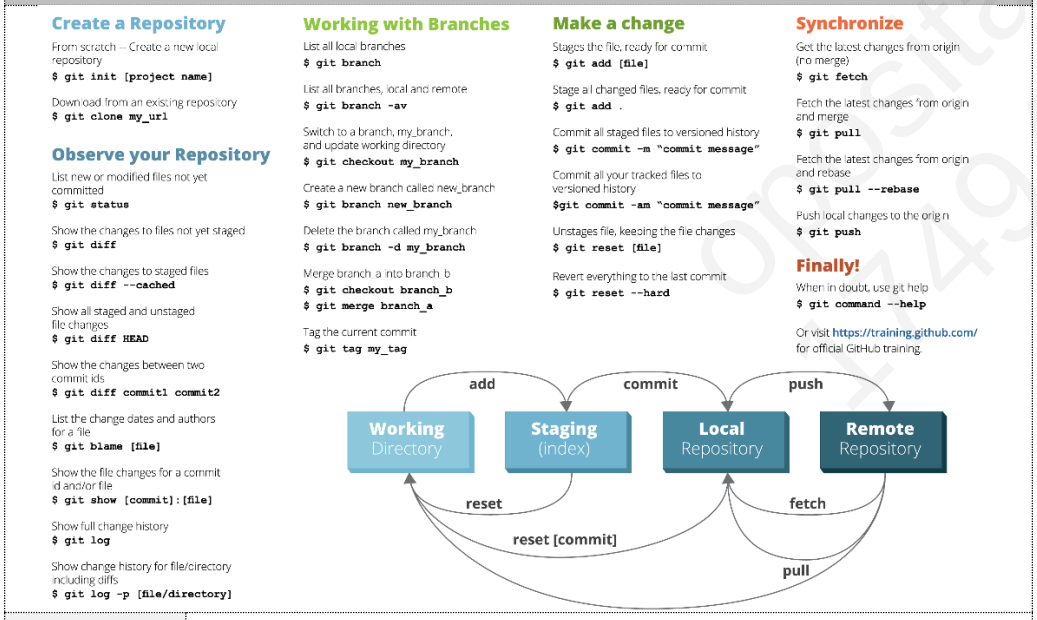
PRUEBAS DEL SOFTWARE	
Pruebas unitarias:	Verificar la funcionalidad de cada componente individualmente una vez que ha sido codificado. (Enfoque caja blanca y caja negra).
De integración:	Verificar el correcto ensamblaje entre los distintos componentes con el fin de comprobar que interactúan correctamente a través de sus interfaces. (Estrategias top-down, bottom-up, combinadas).
Del sistema:	Ejercitar profundamente el sistema comprobando la integración del sistema globalmente, verificando los distintos subsistemas con el resto de sistemas (Tipos: funcionales, comunicaciones, rendimiento, volumen, sobrecarga, disponibilidad de datos, facilidad de uso, operación, entorno y seguridad.
De implantación:	Comprobar el funcionamiento correcto del sistema integrado de hardware y software en el entorno de operación, así como la aceptación del sistema una vez instalado en su entorno real. (Gestión de copias de seguridad y recuperación, pruebas de sobrecarga o de stress).
De aceptación:	Validar que un sistema cumple con el funcionamiento esperado y permitir su aceptación, desde el punto de vista de su funcionalidad y rendimiento. (Pruebas de caja negra que demuestran la conformidad con los requisitos). **Pruebas ALFA**: realizadas en las instalaciones de la organización que desarrolla **Pruebas BETA** (de campo): realizadas por los clientes en sus instalaciones.
De regresión:	Eliminar el efecto onda, es decir, comprobar que los cambios sobre un componente no introducen un comportamiento no deseado o errores adicionales. (implica la repetición de las pruebas que ya se han realizado previamente).
Otros tipos de pruebas:	Las pruebas de humo (**smoke testing**) son una revisión rápida de un producto de software para comprobar que funciona y no tiene defectos evidentes. Las pruebas **fuzzing testing** son un conjunto de pruebas de caja negra que permiten descubrir errores de implementación mediante la introducción de datos al azar, inválidos o malformados.

Herramientas para pruebas del software					
JUnit (unitarias)	Pruebas unitarias para Java	PHPUnit	PHP	xUnit (dirigidas)	.NET
Mockito		CPPUnit	C++	PyUnit	Python
TestNG		NUnit	.NET	QUnit	JavaScript
Selenium	web app	JMeter	Rendimiento	Unit.js	
Cypress		SoapUI	WS en Java	Jasmine	
Badboy		Postman	WS tipo RESTful	Jest	Node.js, React, Angular, Vue
Mocha	Node.js	Cucumber	BDD	FitNesse	Pruebas de aceptación
ATF	Automated Test	Appium	Móviles	EasyAccept	
TestLink	gestión	Robotium	Android	FIT	
UFT (Unified Functional Testing)		funcionales y de regresión			

ISO/IEC 29119 Pruebas del software: Estándar internacional para pruebas de software. Se organiza en 4 partes: (1) Conceptos y definiciones, (2) Procesos, (3) Documentación y (4) Técnicas.

PROGRAMAS PARA CONTROL DE VERSIONES	
Un Sistema de Control de Versiones (SCV o VCS2) es una aplicación que permite gestionar los cambios que se realizan sobre los elementos de un proyecto o repositorio.	
Arquitectura LOCAL:	Los cambios son guardados y usados en un equipo local. Sin colaboración.
Arquitectura CENTRALIZADA o cliente/servidor CVCS (Centralized VCS)	Tienen un único servidor (repositorio) y varios clientes. CVS o Subversion (SVN) de código abierto. Azure DevOps Server (antes TFS), ClearCase o Perforce de código propietario.
Arquitectura DISTRIBUIDA DVCS (Distributed VCS):	Cada cliente posee una réplica del repositorio. Los repositorios intercambian y fusionan revisiones entre sí. Permite trabajar sin conexión. Git, Mercurial (Python), Bazaar, Monotone (C++), Darcs (Haskell), Plastic SCM (Codice Software).

GIT



Integridad	Mediante una suma de comprobación (checksum) hash SHA-1 (40 chars hex).		
Estados	Confirmado (committed)	Preparado (staged)	Modificado (modified)
Secciones	working directory	staging área (Index)	.git directory (Repository)

PLATAFORMAS DE DESARROLLO COLABORATIVO DE SOFTWARE	
Una **forja** es una plataforma de desarrollo colaborativo de software entre desarrolladores.	
GitLab:	Servicio web open source de control de versiones y desarrollo de software colaborativo basado en Git.
GitHub:	Servicio que permite almacenar repositorios Git en la nube.
Bitbucket:	servicio de almacenamiento y colaboración de código basado en Git.
Forja del CTT (Centro de Transferencia de Tecnología)	Creado en GitHub permite que las Administraciones Públicas pueden crear y gestionar sus repositorios.

Otros términos	
DevOps	Conjunto de prácticas que automatizan los procesos entre los equipos (colaboración) de desarrollo de software y TI.
DevOpsSec	Implica la participación de un nuevo actor, el departamento de seguridad.
CI (Continuous Integration)	La integración continua (CI) generalmente incluye la compilación del código, la ejecución de tests automáticos, la generación de artefactos y algunas otras fases como el análisis de calidad de código automático.
CD (Continuous Deployment)	El **despliegue continuo (CD)** es la capacidad llevar cambios al entorno de producción.
CD (Continuous Delivery)	La **entrega continua (CD)** automatiza el proceso de entrega/despliegue en un entorno (pre) con pruebas de aceptación.