SM7A1: Programanció en entorn servidor. Llenguatge PHP

TEMARI

BLOC 1 - CONCEPTES
1- Característiques de la programació en entorn servidor (Pàgines 2 a 4)

BLOC 2 - SISTEMA
1- Entorns IDE (Pàgina 5)
2- Instal·lació de PHP + Apache sobre Linux (Pàgina 5)
3- Instal·lació de PHP + Apache sobre Windows (Pàgina 5)
4- Treballant amb PHP + Apache a partir de contenidors amb Docker (Pàgines 5 a 7)
5- Treballant amb un servidor creat amb l'interpret de PHP des del CLI. (Pàgina 7)

BLOC 3 - LLENGUATGE PHP
1- Estructures de control seqüèncial, condicional i iteratives (Pàgina 8)
2- Treballant amb formularis HTML. Mètodes GET i POST (Pàgina 8)
3- Funcions (Pàgina 8)
4- Tipus de dades i variables. Operadors (Pàgines 9 a 12)
5- Manipulació de fitxers (Pàgina 13)
6- Sessions i Cookies (Pàgina 13)
7- Headers i buffering (Pàgina 13)
8- Objectes i classes (Pàgina 13 a 14)
9- namespace i use (Pàgina 14)
10- Frameworks (Pàgina 14)

BLOC 4 - ANNEXOS
1- Instal·lació de servidor de correu electrònic Postfix (Pàgina 14)
2- PHPMailer (Pàgina 14)
3- Hash de passwords en PHP (Pàgina 14)

BLOC 1- CONCEPTES

1- Característiques de la programació en entorn servidor

a) El desenvolupament d'aplicacions web consisteix bàsicament en crear aplicacions que:

- Utilitzen un navegador que s'executa a l'ordinador de l'usuari i que actua com a interfície entre l'usuari i l'aplicació.
- Un servidor web que generalment es troba en un altre ordinador i amb el qual l'usuari en algún moment s'ha de comunicar pel motiu que sigui com a part del funcionament normal de l'aplicació. Com que l'usuari no podrà comunicar-se directament amb el servidor, ho haurà de fer per mitjà del navegador.
- Un protocol de comunicacions que permet que es comuniquin el servidor i el navegador.

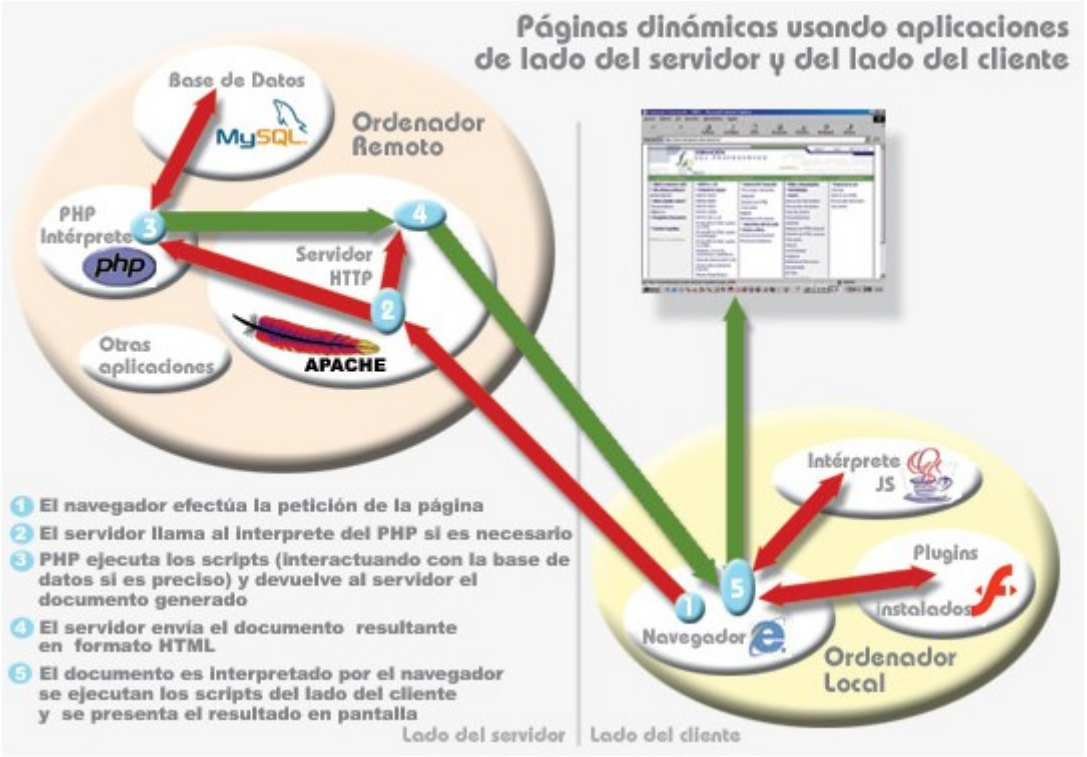
b) Aquest esquema de funcionament permet que el codi que s'executa quan un usuari accedeix a l'aplicació pot estar corrent una part sobre la màquina client a on hi ha el navegador i una altra part en la màquina normalment remota a on s'executa el programa servidor web.

c) Un servidor web és un programa que només té la funció de servir pàgines web (HTML) a un navegador quan aquest fa una petició generada per un usuari, i per tant, no pot executar codi ni utlitzar les dades que un usuari pot haver enviat utilitzant, per exemple, un formulari.

d) Quan un servidor web rep una petició d'executar un codi, passa la petició i les dades rebudes a un altre programa que normalment és un interpret d'ordres. Aquest interpret d'ordres executa la part del codi de l'aplicació de la qual és responsable l'ordinador sobre el qual s'executa el servidor web i quna finalitza la seva execució, passa la resposta al servidor web (generalment un fitxer HTML) que simplement l'ha d'enviar de tornada al navegador.

e) Evidentment, quan parlem de programació en entorn servidor ens referim a aquella part de la programació d'una aplicació que s'executa utilitzant un interpret d'ordres en la maquina sobre la qual està en funcionament el servidor web. Aquesta part de l'aplicació i el llenguatge que es faci servir tindrà unes característiques especials com per exemple, la capacitat de poder comunicar-se amb el servidor web.

f) La següent figura és un esquema dels punts comentats anteriorment:



g) Executar part del codi d'una aplicació en un servidor pot tenir moltes avantatges:

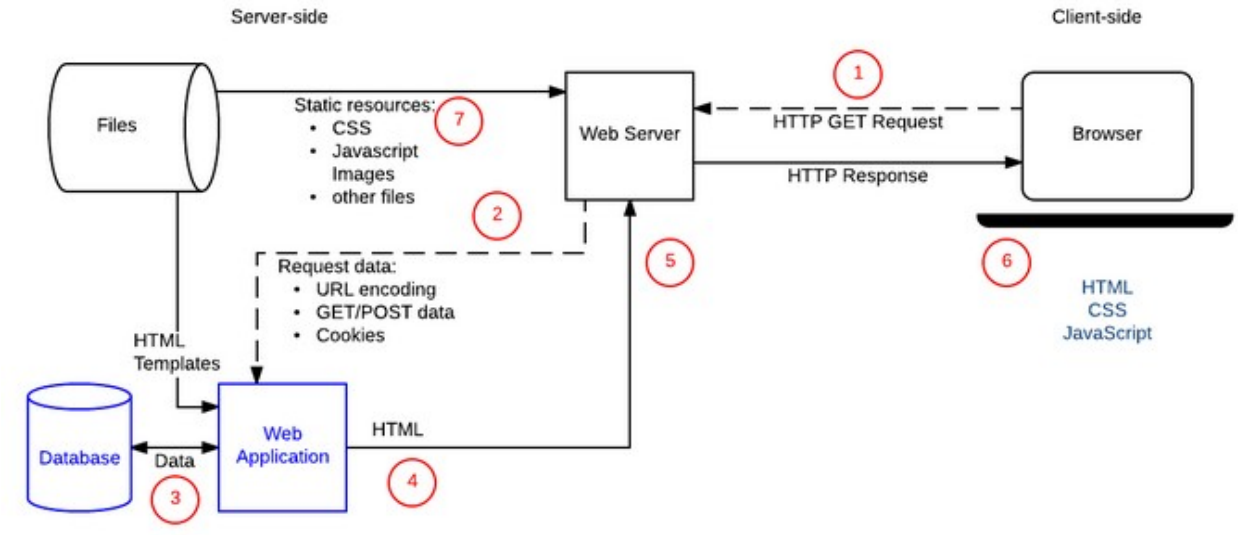
- Permetre l'accés dels usuaris de manera controlada a una base de dades que es troba a l'ordinador servidor de manera que l'usuari no hagi d'executar un servidor de bases de dades de manera local.
- Com a conseqüència del punt anterior, l'usuari tampoc cal que tingui les dades a les qual vols accedir emmagatzemades al seu ordinador.
- Millora de la seguretat
- Creació de contingut web dinàmic i personalitzat en funció de l'usuari, la petició realitzada, les seves preferències i els seus hàbits.
- Facilita emmagatzemar informació per ser reutilitzada posteriorment de manera més ràpida
- Permet executar codi en màquines més potents que la màquina client

h) La programació en entorn servidor es diferencia de la client perquè aquesta darrera s'executa en el costat a on s'executa el navegador. De fet, en la programació en entorn client, el navegador és el responsable d'interpretar el codi.

i) Els navegadors i els servidor web fan servir el protocol HTTP per comunicar-se. Aquest protocol és obert, massivament utilitzat a escala mundial, està molt ben definit, és fàcil de desenvolupar programes client i servidors basat en el protocol, és multiplataforma, ha estat ampliament testejat durant molts anys. També té alguna desavantatge, per exemple, que és un protocol sense estats i això fa que no pugui implementar per defecte sessions.

j) Per mitjà de HTTP, el navegador i el servidor web es comuniquen enviant missatges de petició del navegador cap el servidor (HTTP request) i missatges de resposta del servidor cap el navegador (HTTP response). La petició conté normalment el recurs al qual es vol accedir (una imatge, un fitxer html, una aplicació, una dada emmagatzemada dins d'una base de dades, etc...). La resposta conté normalment un codi HTML estàtic o dinàmic, imatges, texte.....

k) Una aplicació web típica consisteix en una petició enviada pel navegador que el servidor web rep i passa al programa que s'executa en el servidor que pot necessitar accedir a bases de dades o al sistema de fixers del servidor i que quan finalitza la seva execució produeix un codi HTML que es reenvia al servidor web, i aquest de tornada al navegador que mostrarà la resposta per pantalla a l'usuari. Un esquema d'aquest funcionament es pot veure a continuació:



l) El propòsit del llenguatge del costat servidor en general té el propòsit de seleccionar i adaptar el contingut que es retorna a l'usuari, permet treballar amb sessions (molt important), valida dades rebudes, accedeix a bases de dades, controla l'accés als continguts, es responsable de realitzar tasques complexes que podrien relentitzar el funcionament del navegador o creació de contigut dinàmic. En general els servidors són equips amb més potència de càlcul, i tenen dispositius d'emmagatzematge, RAM i connexions de xarxa més ràpides. El propòsit del costat client és mostrar la resposta del servidor a l'usuari millorant la seva apariència i usabilitat.

m) En general el codi del costat client està escrit en llenguatges com HTML, CSS i JavaScript. El codi del costat servidor utilitza llenguatges com PHP, Java, Node.js, C#, Python, Ruby o ASP.

n) PHP és l'acrònim de Hipertext Preprocesor. És un llenguatge de programació del costat del servidor molt popular, gratuït, independent de plataforma, ràpid, amb una gran llibreria de funcions i molta documentació.
o) Avantatges de PHP:

- Fàcil d'aprendre i de realitzar desenvolupament
- Es relativament ràpid perquè no utilitza massa recursos del sistema.
- Suporta en certa mesura l'orientació a objecte. Classes i herència.
- És un llenguatge multiplataforma: Linux, Windows, entre uns altres.
- Capacitat de connexió amb la majoria dels manejadores de base de dades: MysSQL, PostgreSQL, Oracle, MS SQL Server, entre unes altres.
- Capacitat d'expandir el seu potencial utilitzant mòduls.
- Posseeix documentació a la seva pàgina oficial la qual inclou descripció i exemples de cadascuna de les seves funcions.
- És lliure, per la qual cosa es presenta com una alternativa de fàcil accés per a tots.
- Inclou gran quantitat de funcions.
- No requereix maneig detallat del baix nivell.
- Estabilitat

p) Desavantatges de PHP:

- No delega cap tasca en el costat client, de manera que tota la feina la realitza el costat servidor.
- No és adequat per grans aplicacions o una gran número d'apicacions a diferència de per exemple Java.
- Gestió d'errors.
- Llenguatge feblement tipat (ha millorat força i tampoc està clar si es pot considerar o no una desaventatge).

BLOC 2- SISTEMA

1- Entorns IDE

VSCode: https://code.visualstudio.com/
Geany: https://www.geany.org/
PHPStorm: https://www.jetbrains.com/phpstorm/
Netbeans: https://netbeans.org/
Atom: https://atom.io/
Eclipse: https://www.eclipse.org/downloads/

2- Linux Debian 12.7.0 + Apache2.4.64 + PHP8.2.20 + MariaDB10.11.6 (MySQL server)

a) Instal·lació Linux Debian 12.7.0: http://www.collados.org/asix1/sm1/sm1act01.html

b) Instal·lació d'Apache2 + PHP8.2 → Executa com a root o amb sudo:

aptitude update
aptitude install mariadb-client mariadb-server
aptitude install php8.2 php8.2-mysql
aptitude install apache2 apache2-doc libapache2-mod-php

c) Per accedir al client de MariaDB cal executar: **sudo mysql**

d) Els fitxers .php, .js, .css i .html s'han de desar a /**var/www/html**

3- Windows 10/11 + Apache2 + PHP + MySQL

Opció1) Instal·lació de WAMP:

a) Tutorial: https://www.shaileshjha.com/how-to-install-wamp-server-on-windows-10/
NOTA 1: Assegureu-vos que Windows Defender permet a Apache2 tenir connexió a xarxes privades.
NOTA2: Si us falta alguna DLL, descarregueu-la des de https://ww.dll-files.com i deseu-la a **c:\windows\system32**.
b) El fitxers PHP s'han de desar a **c:\wamp64\www**

Opció2) Instal·lació de XAMPP:

a) Tutorial i descarrega: https://www.apachefriends.org
NOTA 1: Assegureu-vos que Windows Defender permet a Apache2 tenir connexió a xarxes privades.
b) El fitxers PHP s'han de desar a **c:\xampp\htdocs**

4- Instal·lació de Docker. Treballant amb contenidors Docker

a) Breu introducció a Docker:

- Llegeix d'aquest enllaç la secció Compairing Docker Containers with virtual machines.
- Mira d'aquest video de youtube des de 0:00 a 2:42
- Documentació oficial de Docker: https://docs.docker.com/get-started/
- Sobre Docker Hub: https://docs.docker.com/docker-hub/
- CLI de Docker: https://docs.docker.com/engine/reference/commandline/cli/

b) Instal·lació de Docker sobre Debian Linux. Executa com a root o amb sudo:

```
apt-get update

apt-get install apt-transport-https ca-certificates curl gnupg2 software-
properties-common

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian
$(lsb_release -cs) stable"

apt-get update

apt-get install docker-ce docker-ce-cli containerd.io docker-compose
```

c) Permet al teu usuari treballar amb dockers. Executa com a root o amb sudo:

```
gpasswd -a $USER docker   (Fes log out i log in perquè l'ordre tingui efecte)
```

d) Per més informació sobre la instal·lació de Docker sobre Debian Linux:
- https://docs.docker.com/install/linux/docker-ce/debian/
- https://docs.docker.com/install/linux/linux-postinstall/#manage-docker-as-a-non-root-user

e) Instal·lació de Docker sobre Windows:
- Informació: https://docs.docker.com/docker-for-windows/install/
- Video de youtube: https://www.youtube.com/watch?v=GIMExUnjzMw
- Mira la secció System Requirements a l'apartat What to know before you install.
- Crea un compte de Docker Hub a https://hub.docker.com/
- Descarrega Docker for Windows Installer.exe des de https://hub.docker.com/?overlay=onboarding
- Executa Docker for Windows Installer.exe i segueix les instruccions que surten per pantalla. Comprova que es generat la icona de Docker al escrptori i que si l'executes s'afegeix una icona de Docker a la barra d'estat (barra inferior dreta).

f) Ara podem descarregar una imatge de Docker amb Apache2 i PHP8.2 des dels dipòsits de Docker Hub i despreś crear un contenidor a partir de la imatge:

- `docker pull php:8.2-apache`

  Descarrega la imatge **`php:8.2-apache`** dels dipòsits de **Docker Hub**

- `docker images`

  Mostra el llistat d'imatges descarregades

- `docker inspect php:8.2-apache`

  Inspecciona les característiques de la imatge. En aquest cas és interessant comprovar el directori de treball de l'apache2 i els ports exposats.

- `docker run --name daw2m07uf1 -p 8080:80 -v ~/m07uf1:/var/www/html -i -t -d php:8.2-apache`

  Executa la imatge **`php:8.2-apache`** i crea un contenidor de nom **daw2m07uf1** amb **apache2** i **php8.2**. Amb l'opció **-p** El port **80** del contenidor és accesible via port **8080** des de màquina host. Amb l'opció **-v**, la carpeta **/var/www/html** del contenidor és accessible via la carpeta **~/m07uf1** de la màquina host. Amb l'opció **-d** el contenidor s'executa en **2n terme** i no ocupa el terminal. Les opcipns **-i -t** permeten accedir al contenidor per mitjà del **bash** (si **bash** està disponible dins del contenidor).

  NOTA 1: La carpeta **m07uf1** es convenient crear-la abans d'executar l'ordre per no tenir problemes de permisos i propietaris.
  NOTA 2: Es pot crear més d'un contenidor amb noms, identificadors i funcionament diferents a partir d'una mateixa imatge

- `docker ps -a`

  Mostra els contenidors en execució, el seu estat d'execució, el seu identificador de contenidor i algune paràmetres més.

g) Comprovació del funcionament del docker creat: Accedeix i treballa amb a la imatge de docker instal·lada a l'apartat anterior. Dins de **~/m07uf1** crea un fitxer de nom **test.php** amb el següent contingut:

```
<?php
      phpinfo();
?>
```

Des del navegador del teu equip accedeix a **http://localhost:8080/test.php**. Comprova que surt una web amb la configuració de PHP del contenidor. Comprova que la versió de PHP del contenidor és la **7.4.**

g) Comprova que pots accedir al bash del contenidor executant:

- `docker exec -it daw2m07uf1 bash`

h) Comprova que amb les ordres:

- `docker stop daw2m07uf1` --> Atures el contenidor i l'aplicació no funciona.
- `docker start daw2m07uf1` --> Poses en marxa el contenidor i l'aplicació funciona

També es pot utilitzar el **CONTAINER ID** per aturar i posa en marxa un contenidor.

i) Per aturar un contenidor i esborrar-lo, executa:

- `docker stop daw2m07uf1` --> Atura
- `docker rm daw2m07uf1` --> Esborra el contenidor
- `docker ps -a` --> Comprova que ja no hi és el contenidor

j) Per esborrar una imatge, primer de tot, cal que no ha existeixi cap contenidor en marxa o aturat associat a la imatge, i després executar:

- `docker rmi php:8.2-apache` --> Esborra `php:8.2-apache`
- `docker images` --> Comprova que ja no hi és la imatge

k) Avantatges i desavantatges de Dockers:
- Avantages Màquines Virtuals: https://www.geeksforgeeks.org/advantages-of-virtual-machines-over-portable-containers/
- Avantatges Dockers: https://www.netapp.com/devops-solutions/what-are-containers/

5- Treballant amb un servidor creat amb l'interpret de PHP des del CLI

a) Es pot crear fàcilment un servidor limitat però que pot permetre executar codi PHP amb l'interpret de PHP des del CLI. Per exemple, si dins de la carpeta **/var/www/html** executem:

```
php -S localhost:8080
```

llavors, qualsevol script .php, .js, .css, .html, etc.. que es trobin dins de **/var/www/html** serà accessible des del navegador si fem una connexió al port **8080** de **localhost**. També es podrà accedir al contingut de qualsevol carpeta que es trobi dins **/var/www/html**.

b) Podem canviar localhost per una adreça IP o un nom de domini.

c) Si executem l'ordre des de qualsevol altra carpeta llavors els fitxers .php, .js, .css, .html, etc.. haurien de trobar-se dins de la carpeta des de la qual hem executat l'ordre.

d) Per més informació: https://www.php.net/manual/en/features.commandline.webserver.php

BLOC 3- LLENGUATGE PHP

1- Estructure de control iteratives i condicionals

a) http://php.net/manual/en/language.control-structures.php
b) if, if..else: http://www.w3schools.com/php/php_if_else.asp
c) switch: http://www.w3schools.com/php/php_switch.asp
d) while i do..while: http://www.w3schools.com/php/php_looping.asp
e) for i foreach: http://www.w3schools.com/php/php_looping_for.asp
f) Array associatiu i foreach: Exemples de treball amb arrays associatius i foreach
g) break: http://php.net/manual/en/control-structures.break.php
i)continue: http://php.net/manual/en/control-structures.continue.php

2- Treballant amb Formularis HTML. Mètodes GET i POST

a) http://www.html-form-guide.com/html-form/how-to-make-a-form.html
b) https://www.safaribooksonline.com/library/view/xforms-essentials/0596003692/ch01s02.html
c) Exemples de formularis i mètodes GET/POST
d) Exemple de formulari que crida a codi que està al mateix fitxer i funció header.
e) CRUD i Mètodes: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

3- Funcions

a) Definició. Tipus. Avantatges: https://www.geeksforgeeks.org/php-functions/
b) Avantatges: http://www.collados.org/daw2/m07/uf1/docs/avantatges.pdf
d) Tipus de funcions:
    User defined --> Definides per l'usuari
    Internal (built-in) --> Del llenguatge (incorporades)
e) Creació (Definició) i utilització bàsica d'una funció:
    http://www.tutorialspoint.com/php/php_functions.htm (Creating PHP Function)
    https://zetcode.com/php/function/ (Defining functions)
f) Característiques: http://php.net/manual/en/functions.user-defined.php --> Definició, Llocs a on pots apareixer, regles per posar noms, necessitat de ser definides abans de ser referenciades,aniament, abast de la definició d'una definició de funció, sobrecarrega, recursivitat.
g) Retorn (opcional) de valors:
    http://php.net/manual/en/functions.returning-values.php
    http://zetcode.com/php/function/ --> (The return keyword)
    http://www.tutorialspoint.com/php/php_functions.htm --> (PHP functions returning value)
h) Pas de paràmetres (o arguments) a funcions:
    http://php.net/manual/en/functions.arguments.php
    http://www.tutorialspoint.com/php/php_functions.htm
    http://zetcode.com/php/function/ → (Function arguments)
i) Exemples de pas de paràmetres: http://www.collados.org/daw2/m07/uf1/exemples/parametres.tar.gz
j) Abast (scope) d'una variable --> Variables globals, locals i estàtiques:
    http://zetcode.com/php/function/ --> (Global and local variables)
    http://php.net/manual/en/language.variables.scope.php
k) Funcions variables i funcions amb longitud variable de paràmetres:
    www.tutorialspoint.com/php/php_functions.htm --> (Dynamic Functions call)
    http://php.net/manual/en/functions.arguments.php --> (Variable-length argument lists)
l) Exemples de funcions variables i funcions amb longitude variable de paràmetres:
    http://www.collados.org/daw2/m07/uf1/exemples/funcvar.tar.gz
m) Llistat de funcions definides en PHP per defecte
    https://www.php.net/manual/en/funcref.php

4- Tipus de dades i variables

a) Llenguatges de programació Strongly typed, weakly typed, statically typed typed i dynamically typed:

- Lectura recomanda: https://android.jlelse.eu/magic-lies-here-statically-typed-vs-dynamically-typed-languages-d151c7f95e2b

- Es considera un llenguatge de programació amb assignació de tipus fort o fortament tipat (en anglès strongly typed) a aquell que obliga al desenvolupador a seguir unes regles molt estrictes d'assignació de tipus a les variables.

- Utilitzant un llenguatge fortament tipat, és força probable que es produeixen errors de compilació o interpretació per aquest motiu. L'assignació de tipus afecta sobre tot a la declaració de variables, funcions, el retorn de dades d'una funció i la crida a funcions.

- Els llenguatges fortament tipat tenen regles estrictes de conversió de tipus.

- Es considera un llenguatge de programació amb assignació de tipus feble o feblement tipat (en anglès weakly typed o loosely typed) a aquell a on pràcticament no existeixen unes regles d'assignació de tipus a les variables que sigui necessari seguir.

- Utilitzant un llenguatge d'assignació de tipus feble és força improbable que es produeixen errors de compilació o interpretació per aquest motiu.

- El llenguatges de programació feblement tipats són més flexibles, ràpids i fàcils d'utilitzar però a vegades es poden produir errors d'execució inesperats per aquest motiu. Els fortament tipats són més estructurats, eviten errors d'execució més fàcilment i els tipus ajuden a fer el codi més manegable quan es treballa en equip.

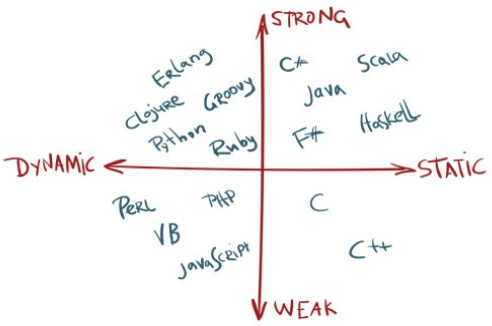
- Un llenguatge d'assignació estàtica de variables, demana que es declari el tipus de variable abans d'utilitzar-la. Quan una variable té un tipus assignat no es pot assignar a a la variable una dada d'un tipus diferent al seu. Si es fa, es produirà un error en temps de compil·lació. Aquests llenguatges permeten trobar més errors durant la fase de desenvolupament i en general donen lloc a executables més ràpids i que requereixen menys memòria.

- Un llenguatge d'assignació dinàmica de variables:
  - No demana que es declari el tipus de variable abans d'utilitzar-la. El tipus es comprova en temps d'execució.
  - En temps d'execució una variable pot treballar amb diferents tipus de dades.
  - Aquest llenguatges són més flexibles, generalment són interpretats (no cal compilació) de manera que el cicle de desenvolupament és més curt, el codi és habitualment menys òptim i poden produir errors durant le'execució.

- PHP o javascript són llenguatge feblement tipats. Java o C# són llenguatges fortament tipats.

- C o Java són llenguatges amb assignació estàtica de variables. PHP o Python treballen amb assignació dinàmica de variables.

- Representació de la situació dels llenguatges en funció del seu nivell dins de les escales d'assignació dinàmica-estàtica i fortament-feblement tipats



b) Tipus de les dades

b) PHP eś un llenguatge feblement dinàmicament tipat però té tipus. Per PHP 7, els tipus són:

- **int:** Número no decimal entre -2.147.483.648 and 2.147.483.647. Es poden utilitzan les bases decimals, hexadecimal, octal i binària.
- **float o double**: Números amb decimals (14) o en forma exponencial (fins a 1,8 e^{308}).
- string: Cadena de caràcters entre " o '.
- **bool:** Una variable que pot valer **true** o **false**.
- **array:** Pot emmagatzemar multiples valors diferenciats. dins d'una variable. Ja han estat tractats a classe.
- **Objecte:** Un tipus de variable que pot emmagatzemar dades i els mètodes (o funcions) per tractar les seves dades.
- **NULL:** Una variable de tipus NULL només pot tenir el valor **null** i això vol dir que no té res assignat, que no té cap valor.

Lectura recomanada: https://www.w3schools.com/php/php_datatypes.asp

c) Abast de l'existència de les variable:

- Una variable és d'**abast local** si es declara dins d'una funció. Només existeix mentre s'executa la funció, i després deixa d'existir. Es pot repetir el nom d'una variable local dins de diferents funcions.

- Una variable és d'**abast global** si es declara fora de qualsevol funció. Exiteix mentre s'executa el codi del script PHP. Si una variable local té el mateix nom que una global, mentre s'executa la funció el valor que s'utilitza és el de la variable local.

- S'utitliza la **paraula clau global** per fer que una variable dins d'una funció sigui no local. Exemple:

- ```
  <?php
        $x = 5;
        function myTest() {
              global $x;
              $x = $x + 2;
        }
        myTest();
        echo $x; // resultat --> 7
  ?>
  ```
- S'utitliza la **paraula clau static** per fer que una variable local no sigui esborrada un cop la funció on va ser declarada finalitza. La propera vegada que es cridi a la funció es podrà utilitzar el valor desat a la variable. Exemple:

  ```
  <?php
  function myTest() {
     static $x =  0;
     echo $x."<br>";
     $x++;
  }
  myTest(); // resultat --> 0
  myTest(); // resultat --> 1
  myTest(); // resultat --> 2
  echo $x."<br>"; //resultat --> Undefined variable
  ?>
  ```

d) Regles de definició de noms de variables:

- Les variables comencen amb $
- Una variable comença amb una lletra o _
- Una variable no comença mai amb un número
- Una variable només pot tenir els caràcters [0..9], [a..z], [A..Z] i _
- Les variables són consideren diferents les majúsucules i les mnúcules. $temp és diferent a $Temp.
  ```

e) Declaració de tipus en el pas de paràmetres a funcions. Declaració estricta i no estricta:

- PHP7 permet declarar el tipus dels arguments que passem a una funció quan declarem la funció . En cas de no complir-se el tipus, l'interpret fa un canvi de tipus. Així doncs, en PHP7 es pot definir una funció de la següent manera:

```php
<?php
function myAdd(int $a, int $b, int $c) {
   return $a + $b + $c;
}
echo myAdd(4.2,3,-2.7); // resultat --> 5 (no 4.5)
?>
```

- Es pot obligar a que les dades que es passin a la funció siguin obligatoriament del mateix tipus que les demanades a la declaració indicant al principi del script PHP que les declaracions seran **estrictes**. En aquest cas, durant l'execució es produirà un error si una variable no ś del tipus demana. Això es pot fer de la següent manera:

```php
<?php
declare(strict_types=1);
function myAdd(int $a, int $b, int $c) {
   return $a + $b + $c;
}
echo myAdd(4,3,-2); // resultat 5
# echo myAdd(4.2,3,-2); // resultat --> Error. Pàgina en blanc
?>
```

Lectura recomanada: https://blog.teamtreehouse.com/5-new-features-php-7

f) Retorn de funcions amb declaració de tipus:

- PHP7 permet declarar el tipus de la variable retornada:

```php
<?php
function myAdd(float $a, float $b, float $c) : int {
   return $a + $b + $c;
}
echo myAdd(7.1,3.2,-2.7); // resultat --> 7 (no 7.6 )
?>
```

Lectura recomanada: https://blog.teamtreehouse.com/5-new-features-php-7

g) Operadors amb variables:

- Operadors aritmètics: + * - / ** %
- Operadors de comparació: ==, !=, <=, >=, <, >, ===, !== , <=>
- Operadors d'assignació: =, +=, -=, *=, %=
- Operadors lògics: && (and), || (or), xor, !,
- Operadors incrementals i decrementals: ++ i --
- Operadors amb strings: . (concatena) .= ($a .= $b ---> $a=$a.$b)
- Operadors amb arrays: +, ==, ===, !=, <>, !==

Lectura recomanda: https://www.w3schools.com/php/php_operators.asp

h) Operadors ternaris:

- L'opeardor ternari és un operador condicional que disminueix la longitud del codi mentre realitza comparacions i condicionals.
- Aquest mètode és una alternativa a utilitzar sentències if-else i if-else niutats.
- L'ordre d'execució d'aquest operador és d'esquerra a dreta.

```php
<?php
$nota = 6.6;
$missatge = ($nota >= 5) ? "Aprovat" : ""Suspés";
// Si $nota >=5 $missatge="Aprovat" sino $missatge="Suspés";
echo $missatge // resultat --> Aprovat
?>
```

i) Casting (Conversió) de tipus variables:

- Podem forçar una variable a utilitzar un determinat tipus utilitzant el casting (forçat) de tipus. Per exemple:

```php
<?php
$b = 4.2;
$a = (int) $b; // El tipus de $b serà integer o sencer
gettype($a).”<br>”;  // Resultat: int (integer)
$gettype($b).”<br>”; // Resultat: float o double (real)
?>
```

- Tipus bàsics de casting: (int), (float) o (double), (string), (bool), (array) i (object)

- Es pot obtenir el tipus d'una variables amb la funció gettype() i es pot canviar el tipus d'una variable amb la funció settype().

- També es pot comprovar el tipus d'una variable amb les funcions is_int($var), is_string($var), is_bool($var), is_numeric($var), is_null($var), is_string($var) i is_array($var)

j) Variables predefinides o reservades:

- PHP 7 té una sèrie de variables predefines (i per tant reservades) i els desenvolupadors no poden definir variables que tinguin el mateix nom que una variable predefinida.
- Llista de variables predefinides: http://php.net/manual/en/reserved.variables.php

k) Taules de comparació de tipus: http://php.net/manual/en/types.comparisons.php

l) Funcions per visualitzar els continguts i les estructures dels tipus:

- Es convenient saber utilitzar les funcions echo, print, var_dump i print_r
- Documentació: http://cybmeta.com/php-diferencias-entre-echo-print-print_r-y-var_dump

m) Arrays:

- Definició i creació de arrays: https://www.w3schools.com/php/php_arrays.asp
- Array indexat: https://www.w3schools.com/php/php_arrays_indexed.asp
- Arrays associatiu: https://www.w3schools.com/php/php_arrays_associative.asp
- Array multidimensional: https://www.w3schools.com/php/php_arrays_multidimensional.asp
- Exemples: http://www.collados.org/daw2/m07/uf1/exemples/arrays.tar.gz

5- Manipulació de fitxers

a) Funcions per treballar amb fitxers:
http://php.net/manual/en/ref.filesystem.php
b) Funcionament de les funcions de manipulació de fitxers més importants:
- Obrir/llegir/tancar: http://www.w3schools.com/php/php_file_open.asp
- Crear/Escriure: http://www.w3schools.com/php/php_file_create.asp
- Pujar un fitxer: http://www.w3schools.com/php/php_file_upload.asp

c) Funció explode.
d) Exemples-I: Exemples de manipulació de fitxers
f) Exemples-II: Exemple de gestió d'usuaris d'una aplicació per mitjà de fitxers de texte.
e) Manipulació de directoris: https://www.w3schools.com/php/php_ref_directory.asp

6-Sessions i cookies

a) Concepte de cookie: https://en.wikipedia.org/wiki/HTTP_cookie
b) Consideracions importants sobre les cookies: http://www.collados.org/daw2/m07/uf1/docs/cookies.pdf
c) Creació, recuperació, modificació, eliminació, comprovació de la possibilita de treballar amb cookies: http://www.w3schools.com/php/php_cookies.asp
d) Funció setcookie - I: http://www.w3schools.com/php/func_http_setcookie.asp
e) Funció setcookie - II: http://php.net/manual/en/function.setcookie.php
f) Sessions - I: Concepte de sessió, $_COOKIE, $_SESSION i directives de php.ini
g) Session - II: https://code.tutsplus.com/tutorials/how-to-use-sessions-and-session-variables-in-php--cms-31839
h) Treballant amb $_SESSION – I
i) Treballant amb $_SESSION – II
j) Treballant amb $_SESSION – III
k) Exemples de treballs amb sessions
l) Expiració de sessions

7-Headers i buffering

a) Conceptes de buffering
b) Conceptes sobre headers d'un missatge HTTP i de la funció header()
c) Exemples de header() i buffering

8- Programacio orientada a objectes

a) Conceptes bàsics: http://www.collados.org/daw2/m07/uf1/docs/poo.pdf
b) Documentació sobre OOP per PHP de Jan Bodnar: https://zetcode.com/php/oop/
c) Pàgina oficial de ló de PHP: https://www.php.net/manual/en/language.types.object.php
d) OOP de w3schools.com: https://www.w3schools.com/php/php_oop_what_is.asp
e) Constructors i destructors:
- https://www.zentut.com/php-tutorial/php-constructor-and-destructor/
- https://desarrolloweb.com/articulos/1748.php
- https://www.w3schools.com/php/php_oop_constructor.asp
- https://www.w3schools.com/php/php_oop_destructor.asp

f) Modificadors d'accés a atributs
- https://www.w3schools.com/php/php_oop_access_modifiers.asp

g) Constants:
- https://www.w3schools.com/php/php_oop_constants.asp

h) Mètodes estàtics:
- https://www.w3schools.com/php/php_oop_static_methods.asp

i) Atributs o propietats estàtiques:
- https://www.w3schools.com/php/php_oop_static_properties.asp
- https://www.geeksforgeeks.org/when-to-use-static-vs-instantiated-classes-in-php/ (amb explicació de quan és millor treballar amb mètodes estàtics i quan es millor instanciar un objecte)

j) Interfícies:
- https://diego.com.es/interfaces-en-php
- https://www.geeksforgeeks.org/php-interface/

k) Classes abstractes
- https://www.php.net/manual/es/language.oop5.abstract.php
- https://www.w3schools.com/php/php_oop_classes_abstract.asp

l) Classes abstractes vs Interfícies
- https://www.brandonsavage.net/interfaces-or-abstract-classes/

m) Creant fitxers PDF amb la classe FPDF
- https://www.elated.com/create-nice-looking-pdfs-php-fpdf/
- https://desarrolloweb.com/manuales/manual-fpdf.html

n) Exemples
- Exemples: http://www.collados.org/daw2/m07/uf1/exemples/poo.tar.gz

9- Treballant amb namespace i use

a) Concepte de **namespace**:
- https://www.w3schools.com/php/php_namespaces.asp
- https://www.php.net/manual/en/language.namespaces.importing.php

b) Concepte de **use**: https://www.php.net/manual/en/language.namespaces.importing.php
c) Lectura 1 (especialment la secció "Introducción"): https://diego.com.es/namespaces-en-php
d) Lectura 2 ((Especialment la secció "Why namespaces"): https://www.phptutorial.net/php-oop/php-namespace/
e) Exemples: http://www.collados.org/daw2/m07/uf1/exemples/namespaces_use.tar.gz

10- Frameworks i composer. Framework Dompdf.

a) Concepte de framework i programa composer: http://www.collados.org/daw2/m07/uf1/docs/Framework-Composer-Dompdf.pdf
b) Framework Dompdf: http://www.collados.org/daw2/m07/uf1/docs/Framework-Composer-Dompdf.pdf

BLOC 4 - ANNEXOS

1- Instal·lació de servidor de correu electrònic Postfix

2- PHPMailer

3- Hash de passwords en PHP:
- How to hash passwords in PHP
- Good Practices: PHP Security, How to manage password