

Webmastering II

Partie I : Les Éléments de Base du Langage PHP

Sommaire

1. Variables et Constantes
2. Opérateurs
3. Structures de Contrôle
 - o Structures Alternatives
 - if..else
 - switch()
 - les boucles
4. Tableaux
 - o Numériques
 - o Associatifs
 - o Multi-dimensionnels
5. Chaines et Fonctions de Manipulation
6. Super Globaux
 - o Formulaires
 - o SESSION et Cookies
 - o Variables d'URL
 - o Variables Serveur
 - o Variables Globales
7. Fonctions
8. Dates
9. Inclusion des Pages
- 10.Fichier
- 11.Mini Projet

Chapitre I

Variables ,Constantes et Opérateurs

1) Creation et Execution d'un Script PHP

un fichier source PHP peut contenir du :

- PHP entouré des balises `<?php ?>`
- HTML est le langage par défaut
- CSS entouré des balises
`<style></style>`
- JAVASCRIPT entouré des balises
`<script> </script>`

2) Instructions d'affichage

- `echo()` : affiche une variable de type primitifs ou une constante
- `print()` : affiche une variable de type primitifs ou une constante
- `var_dump()` : affiche une variable ou une constante ainsi que le type

Exemple :

```
<?php  
    echo "Cours de WebMastering Semestre 2"; echo"<br>";  
    echo 'Cours de WebMastering Semestre 2'; echo"<br>";  
    print 'Cours de WebMastering Semestre 2';echo"<br>";  
    print "Cours de WebMastering Semestre 2";echo"<br>";  
    var_dump("Cours de WebMastering Semestre 2") ;  
?>
```

NB: La fonction var_dump() fonction est généralement utilisé pour déboguer une variable

3) Commentaire: permet de documenter le code ,on les commentaires:

- Uniligne noté //
- Multiligne noté /**/

Exemple :

```
/*
```

NB: Une balise HTML qui est affichée dans un bloc PHP doit être précédé par la fonction echo par exemple :
**echo"
";**

```
*/
```

○ **Types**

■ **Elémentaire :**

- int
- float
- bool
- string
- null

■ **Composé**

- Tableau ou Array
- Objet
- Ressource
- File
- Traits

NB : Pour connaître le type d'une variable , on peut utiliser les fonctions `is_type()` ou `gettype()`

4) Convention de nommage

- **Variable**: camelCase

Exemple: `$nomVar`

- **Fichier**: Kebab Case

Exemple: `mon.fichier.php`

- **Fonctions**: Snake Case

Exemple: `nom_fonction()`

- **Constantes**: Snake Case Majuscule

Exemple: `MA_CONSTANTE`

5) Variables : \$nomVar

Exemple : Affichage , Variables et Type

```
<?php

//Variables

    $entier=2;//Entier ou int

    $reel=2.5; //réel ou float

    $chaine1="Bonjour"; //chaine ou string

    $chaine2='Bonjour'; //chaine ou string

    $boolean=true; // boolean

// Affichage avec echo

    echo 'La valeur entière est ' . $entier;echo"<br>";

    echo "La valeur reel est $reel";echo"<br>";

    echo "La valeur chaine est {$chaine1}";echo"<br>";

// Affichage avec print

    print("La valeur chaine est $chaine2");

//Affichage Valeurs et Types

    var_dump($entier);echo"<br>";

    var_dump($reel);echo"<br>";

    var_dump($chaine_1);echo"<br>";

    var_dump($chaine2);echo"<br>";

    var_dump($boolean);

?>
```

NB: L'injection de variables fonctionne uniquement avec

les guillemets "" et pas avec les '';

6) Constantes

define("MA_CONSTANTE", valeur);

Exercice Application :

```
<?php

    define("PI",3.14);

    echo "La valeur est ".PI;

    echo "<br>";

    //Constantes Systemes

    echo 'Separateur de Dossier '.DIRECTORY_SEPARATOR. '<br>';

    //Quelques Constantes Magiques

    echo 'Numéro de ligne : ' .__LINE__. '<br>';

    //Affiche le chemin du fichier et son nom

    echo 'Chemin complet du fichier : ' .__FILE__. '<br>';

    //Affiche le nom du dossier qui contient le fichier

    echo 'Dossier contenant le fichier : ' .__DIR__. '<br>';

    //Affiche à nouveau la ligne où la constante a été appelée

    echo 'Numéro de ligne : ' .__LINE__. '<br>';
```

7) Opérateurs

a) Arithmétiques

`+ , - , * , / , %`

b) Relationnels

`==, ===, !=, <=, >=, <, >`

c) Logiques

`&& ou and, || ou or , !`

d) Affectation (`=`)

e) Affectation Composé

`+=, -=, *=, /=, %=`

f) Incrémentation (`++`)

g) Décrémentation (`--`)

Remarques

- Affichage des variables serveur en utilisant la fonction `phpinfo()`
- En PHP on peut l'affichage des Erreurs sur `php.ini` en changeant les valeurs des ces deux modules comme suit
 - `error_reporting=E_ALL`
 - `display_errors = on`

APPLICATIONS

Exercice I :

Générer deux nombres puis calculer et afficher:

- Somme
- Exponentiel
- Différence
- Produit
- Modulo
- Division
- Carré

TAF ETUDIANT :

Exercice II: Générer deux nombres puis faire leur permutation. On affichera les deux nombres avant et après permutations.

Exercice III : Point: Générer deux points puis calculer et afficher la distance entre les deux points. Un point est caractérisé par son abscisse et son Ordonnée.

Chapitre II

Structures de contrôle

8) Structure conditionnelle

- a) `if(condition) ..else`
- b) `if(condition)..elseif (condition)`
- c) `switch($var) .. cas`

APPLICATIONS

Exercice : Générer un nombre puis déterminer et afficher son signe

Exercice : Générer la valeur d'un mois(entier) puis afficher sa correspondance en chaîne.

TAF ETUDIANT :

Exercice : Générer deux nombres entiers puis les afficher dans l'ordre croissant et dans l'ordre décroissant

Exercice : Générer trois nombres entiers puis les afficher dans l'ordre croissant et dans l'ordre décroissant

Exercice : soit l'équation $ax+b=0$,générer la valeur de a et de b puis donner les solution de l'équation ci dessus.

Exercice : soit l'équation $2+bx+c=0$,générer la valeur de a , de b et de c puis donner les solution de l'équation ci dessus.

9) Structure itérative

- a) for..**
- b) while**
- c) do..while**

APPLICATIONS :

Exercice 1 :

Écrire un script qui génère un nombre supérieur à 10 000 puis affiche dans:

- une table HTML ,les nombres premiers sont compris entre 1 et la valeur entrée.**

TAF ETUDIANT :

Écrire un script qui génère un nombre supérieur à 10 000 puis affiche dans:

- une table HTML , les valeurs sont inférieures à la moyenne du tableau**
- une table HTML ,les valeurs supérieures à la moyenne.**

NB: on utilisera la fonction `set_time_limit()` qui permet d'augmenter la durée d'exécution du script.

Chapitre III

Les Tableaux et Chaînes

A) Les Tableaux

1) Tableau Numérique

- Déclaration
- Initialisation
- Parcours avec la boucle foreach()
- Opérations
 - Ajout
 - Suppression
 - Recherche
 - Modification
- Quelques Fonctions utiles
 - cout()
 - array_push()
 - array_pop()
 - array_shift()
 - array_unshift()
 - in_array()
 - explode()
 - implode()
 - etc.

Exercice Application :

```
<?php
//Déclaration
$array_num=[];
var_dump($array_num);
//Initialisation
$array_num=["Bonjour",'Au Revoir',1,2.5,true];
var_dump($array_num);
echo"Afficher un element du tableau";echo"<br>";
var_dump($array_num[0]); echo"<br>";
var_dump($array_num[4]); echo"<br>";
echo"Parcours du Tableau"; echo"<br>";
foreach ($array_num as $key => $value) {
    echo"{$key}>{$value}"; echo"<br>";
}
echo"Ajout d'un elt en Fin du Tableau"; echo"<br>";
array_push($array_num,"Nouvelle valeur 1");
var_dump($array_num);echo"<br>";
$array_num[]="Nouvelle valeur 2";
var_dump($array_num);echo"<br>";
echo"Ajout d'un elt en Debut du Tableau";
echo"<br>";
array_unshift($array_num,"Nouvelle valeur 1 en debut");
var_dump($array_num);echo"<br>";
echo"Supression d'un elt en Fin du Tableau";
echo"<br>";
$val=array_pop($array_num);
var_dump($array_num);echo"<br>";
echo"Supression d'un elt en Debut du Tableau";
echo"<br>";
$val=array_shift($array_num);
var_dump($array_num);echo"<br>";
```

```

echo"recherche d'une valeur dans le tableau Tableau";
echo"<br>" ;
$val=in_array(1,$array_num) ;
var_dump($val);echo"<br>" ;
echo"recherche d'une clé dans le tableau Tableau";
echo"<br>" ;
$val=key_exists(5,$array_num) ;
var_dump($val);echo"<br>" ;

echo"Recuperation des clés dans le Tableau";
echo"<br>" ;
$val=array_keys($array_num) ;
var_dump($val);echo"<br>" ;
echo"Recuperation des valeurs du Tableau";
echo"<br>" ;
$val=array_values($array_num) ;
var_dump($val);echo"<br>" ;

echo"Conversion Tableau to String"; echo"<br>" ;
$val=implode(":",$array_num) ;
var_dump($val);echo"<br>" ;
//Tableau Multi dimentions
$arr_multi=[

    [1,2,7,8] ,
    [1.5,2.4,8.6,8] ,
];
var_dump($arr_multi);echo"<br>" ;

```

APPLICATIONS:

Exercice 1 :

Écrire un script qui génère une valeur supérieure à 10 000 puis crée des tableaux numériques :

- T1 est associée à l'ensemble des nombres premiers compris entre 1 et la valeur entrée.
- T2 à l'ensemble des valeurs qui sont inférieures à la moyenne du tableau
- T3 à l'ensemble des valeurs qui sont supérieures à la moyenne.

Les tableaux T1 , T2 et T3 sont affichés à l'aide de trois Table HTML.

Règle de Gestion

- **RG 1 : On utilisera des tableaux à une dimension**
- **RG 2 :On utilisera des tableaux à deux dimensions**

2) Tableau Associatifs

- Déclaration
- Initialisation
- Parcours avec la boucle foreach()
- Opérations
 - Ajout
 - Suppression
 - Recherche
 - Modification
- tri
 - bulle
 - insertion
 - fusion
- Quelques Fonctions utiles
 - cout()
 - array_push()
 - array_pop()
 - array_shift()
 - array_unshift()
 - in_array()
 - explode()
 - implode()
 - etc.

Exercice Application :

```
?php

//Declaration

$bien=[

    "id"=>1,

    "reference"=>"Ref001",

    "Description"=>"Lorem ipsum dolor sit amet
consectetur adipisicing          elit. At commodi
quia quis. Odit itaque dicta distinctio cumque
placeat possimus iure, amet fugiat rem at voluptas
eligendi saepe accusantium neque doloribus?",

    "prix"=>500000,

    "zone"=>[

        "id"=> 1 ,

        "nom"=> "zone 1"

    ] ,

    "type"=>"chambre"

] ;

var_dump($bien);

echo "Affichage de la Reference";echo"<br>";

var_dump($bien['reference']);

echo "Affichage du nom de la zone";echo"<br>";

var_dump($bien['zone'][ 'nom']);echo"<br>";
```

```
//Tableau multi dimentionnel

$arr_bien=[

    [
        "id"=>1,
        "reference"=>"Ref001",
        "description"=>"Lorem ipsum dolor sit amet consectetur adipisicing elit. At commodi quia quis. Odit itaque dicta distinctio cumque placeat possimus iure, amet fugiat rem at voluptas eligendi saepe accusantium neque doloribus?",

        "prix"=>50000,
        "zone"=>[
            "id"=> 1 ,
            "nom"=> "zone 1"
        ],
        "type"=>"chambre"
    ],
    [
        "id"=>2,
        "reference"=>"Ref002",
        "description"=>"Lorem ipsum dolor sit amet consectetur adipisicing elit. At commodi quia quis. Odit itaque dicta distinctio cumque placeat possimus iure, amet fugiat rem at voluptas eligendi saepe accusantium neque doloribus?",

        "prix"=>500000,
        "zone"=>[
            "id"=> 1 ,
            "nom"=> "zone 2"
        ],
    ]
]
```

```
"type"=>"Appartement"
]

];

foreach ($arr_bien as $key => $bien) {

    echo "-----Bien {$key}-----<br>";
    echo "ID: {$bien['id']} <br>";
    echo "REFERENCE: {$bien['reference']} <br>";
    echo "DESCRIPTION: {$bien['description']} <br>";
    echo "PRIX: {$bien['prix']} <br>";
    echo "ZONE: {$bien['zone']['nom']} <br>";
}
```

B) Les Chaînes

- Déclaration
- fonctions utiles
 - `strlen()`
 - `substr()`
 - `str_replace()`
 - `strpos()`
 - `explode()`
 - `strip_tags()`
 - `stripslash()`
 - `str_word_count()`
 - `trim()`
 - `ltrim()`
 - `rtrim()`

Exercice Application :

```
<?php

$chaine="Ceci est une chaine de Caractere";
//Affichage du premier caractere
echo "Affichage du premier caractere ". $chaine[0]."<br>" ;
//Modification du premier caractere
echo "Affichage Modification ". $chaine."<br>" ;
$chaine[0]="F";
echo "Apres Modification ". $chaine."<br>" ;
//Quelques Fonctions utiles des Chaines de Caracteres
echo "Quelques Fonctions utiles des Chaines de Caracteres <br>" ;
//Longueur de Chaine
echo "La longueur de la chaine ".strlen($chaine)."<br>" ;
//Extraction d'une partie de la Chaine
echo "Recuperation des 2 premiers caracteres
".substr($chaine,0,2)."<br>" ;
//Remplacer des caracteres de la chaine
echo "Remplacer tous les e par des a
".str_replace("e","a",$chaine)."<br>" ;
//Enlever les espaces en debut et en fin de mot
echo "Enlever les espaces en debut et en fin de mot
".trim($chaine)."<br>" ;
echo "Recherche du mot caractere dans la chaine
".strpos($chaine,"Caractere");
echo "<br>" ;
//Convertir un chaine en tableau
$arr=explode(" ",$chaine);
var_dump($arr);
//Supprimer les balises d'une chaine
$text = '<p>Test paragraph.</p><!-- Comment --> <a
href="#fragment">Other text</a>';
echo strip_tags($text);
echo "<br>" ;
//stripslashes - Supprime les antislashes d'une chaîne
$str = "Avez-vous l'oreille dure?";
echo stripslashes($str);
echo "<br>" ;
//Recherche d'une sous chaine dans une chaine

//Le nombre de Mot de la Chaine
echo "Le nombre de mots d'une chaine
".str_word_count($chaine)."<br>" ;
```

APPLICATIONS :

Exercice 1 : Générer des numéros ayant le format suivant LLL-CCC-LCL.

NB : L signifie Lettre et C signifie chiffre.

TAF ETUDIANT

Exercice 2 :

Générer une phrase puis écrire un script qui enlève tous les espaces inutiles de la phrase.

Règles de Gestion

- Les espaces inutiles sont:
 - les espaces en début et fin de chaîne
 - les espace avant un point
 - les espaces successives
 - les espaces avant une apostrophe

Exercice 3:

Écrire un script qui permet de générer un tableau N de mots. Chaque mot ne devrait contenir que 20 caractères. Le script affiche tous les mots du tableau puis détermine et affiche ;

- le mot le plus long et le mot le plus court
- le nombre de mots contenant la lettre « M » (la casse n'est pas tenue en compte).
- le mot qui a le plus de voyelles
- le mot qui a moins de consonne

CHAPITRE IV

Tableaux Superglobaux

Un tableau 'superglobaux', ou variable globale automatique. Cela signifie simplement que cette variable est disponible dans tous les contextes du script. Il n'est pas nécessaire de faire **global \$variable**; pour y accéder dans les fonctions ou les méthodes.

On distingue les tableaux superglobaux suivants:

- **\$_POST**
- **\$_GET**
- **\$_SESSION**
- **\$_REQUEST**
- **\$_SERVER**
- **\$_GLOBAL**

A) Gestion des Formulaires avec `$_POST`

- `$_POST`
 - Formulaire
 - Fonctions de validation
 - `empty()`
 - `isset()`
 - `is_type()`
 - `filter_var()`

Application I:

Réaliser un script qui effectue des opérations arithmétiques (+, -, /, %, *) à partir de deux nombres entrés à partir d'un formulaire.

Exercice d'application : calculatrice.php

Code HTML

```
<!doctype html>
<html lang="en">
  <head>
    <title>Title</title>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.
min.css"
      integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUhcWr7x9J
voRxT2MZw1T" crossorigin="anonymous">
  </head>
  <body>

    <div class="container mb-5">
```

```

<h2 class="text-info">Calculatrice</h2>
<form>
    <div class="form-group row">
        <label for="inputName" class="col-sm-2
col-form-label"> Nombre 1</label>
        <div class="col-sm-6 ml-2">
            <input type="text" class="form-control"
name="nbrel" id="inputName" placeholder="">
        </div>
    </div>
    <div class="form-group row">
        <label for="inputName" class="col-sm-2
col-form-label"> Nombre 2</label>
        <div class="col-sm-6 ml-2">
            <input type="text" class="form-control"
name="nbre2" id="inputName" placeholder="">
        </div>
    </div>
    <div class="form-group row">
        <label for="" class="col-sm-2
col-form-label">Operateur</label>
        <select class="form-control col-sm-6 ml-3" name="op"
id="">
            <option>Addition </option>
            <option>Soustraction</option>
            <option>Division</option>
        </select>
    </div>
    <div class="form-group row">
        <div class="offset-7 col-sm-4 ">
            <button type="submit" class="btn btn-primary"
name="btn_submit" value="btn_egal">=</button>
        </div>
    </div>

    </form>
</div>
<p class="mt-2">
    <?php echo "<strong>Resultat</strong> :{$result}"; ?>
</p>

<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->

```

```

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8ab
tTE1Pi6jizo" crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper
.min.js"
integrity="sha384-UO2eT0CpHqdsJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF
86dIHNDz0W1" crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.mi
n.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x
0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

Code PHP

```

<?php
    if(key_exists("btn_submit", $_POST)) {
        $arr_error=[];
        $nbrel=
        trim(stripslashes(strip_tags($_POST['nbrel'])));
        $nbre2=
        trim(stripslashes(strip_tags($_POST['nbre2']))));
        $op=trim(stripslashes(strip_tags($_POST['nbre2']))));
        $result="";
        switch($op) {
            case "+":
                $result=$nbrel+$nbre2;
                break;
            case "-":
                $result=$nbrel-$nbre2;
                break;
            case "*":
                $result=$nbrel*$nbre2;
                break;
            case "/":
                if($nbre2!=0)
                    $result=$nbrel/$nbre2;
                else
                    echo "Impossible";
                break;
            case "%":

```

```
    if ($nbre2!=0)
        $result=$nbre1%$nbre2;
    else
        echo "Impossible";
        break;
    break;

}
?>
```

Règles de validation du formulaire:

- Les champs nbre 1 et nbre 2 sont Obligatoires
- Les champs nbre 1 et nbre 2 sont des chaîne

B) Gestion de la Connexion avec \$_SESSION

- \$_SESSION et \$_COOKIES
 - Notions de Session et de Cookies
 - Gestion des Erreurs dans un Formulaire

Application I:

Gestion des Formulaires et de la Session

A) Description du Projet

Réaliser un système de connexion de déconnexion et d'enregistrement de nouveaux comptes .

B) Règles de Gestion:

- RG1: Un compte de connexion est caractérisé par :
 - nom et le prenom
 - login
 - password
 - role (Admin ou Visiteur)
- RG2: Le login est un email
- RG3: Les utilisateurs sont stockés dans la superglobale \$_SESSION
- RG4: les messages d'erreurs de validation seront stockés dans la session.
- RG5 : Après la connexion d'un utilisateur , il est redirigé vers sa page d'accueil, accueil.visiteur.html.php pour le visiteur et accueil.admin.html.php.

C) Règles de Validation:

- RV1: Tous les champs sont obligatoires
- RV2: Le login est unique
- RV3: Lors de l'inscription on doit confirmer le mot de passe

D) Structure du projet

- views
 - login.htm.php
 - register.html.php
 - accueil.visiteur.html.php
 - accueil.admin.html.php
 - show.user.html.php
 - confirmation.html.php
 - index.php
- controllers
 - security.php

C) Gestion des variables d'url avec `$_GET`

- `$_GET`
 - Variables d'url
 - Inclusion de Pages
 - require et require_once
 - include et include_once
 - ob_start()\$var=ob_get_clean()
 - Mise en Place d'un Menu

Application II:

Menu de navigation et Autorisation

A) Mettre en place un menu contenant les items suivants:

- Accueil
- Inscription
- Connexion ou Déconnexion
- Utilisateurs qui permet de lister tous les utilisateurs.

B) Autorisations:

- Un Admin a accès à tous les menus
- Un Visiteur à accès à tous les menus sauf au menu utilisateurs.

NB: Pour les inclusions de pages on utilisera les inclusions simples puis les notions de page de présentation ou layout.

D) La Variable Superglobale **\$_REQUEST**

La variable superglobale **\$_REQUEST** est un tableau associatif qui contient par défaut le contenu des variables **\$_GET**, **\$_POST** et **\$_COOKIE**.

E) La Variable Superglobale **\$_SERVER**

\$_SERVER est un tableau contenant des informations comme les en-têtes, dossiers et chemins du script. Les entrées de ce tableau sont créées par le serveur web.

Exemple :

```
<?php  
  
// Methode d'envoie d'envoi de donnée GET ou POST  
  
echo "Methode d'envoi des données ". $_SERVER["REQUEST_METHOD"] ;  
  
echo "<br>" ;  
  
//Adresse du serveur  
  
echo "adresse du serveur ". $_SERVER["HTTP_HOST"] ;  
  
echo "<br>" ;
```

```
//PHPSESSID ou identifiant de connexion du Serveur

echo "PHPSESSID ou identifiant de connexion du Serveur " .
$_SERVER["HTTP_COOKIE"] ;

echo "<br>";

//Nom du Serveur

echo "Nom du Serveur ". $_SERVER["SERVER_NAME"] ;

echo "<br>";

//Port de Connexion

echo "Port de Connexion ". $_SERVER["SERVER_PORT"] ;

echo "<br>";

//Document Racine

echo "Document Racine ". $_SERVER["DOCUMENT_ROOT"] ;

echo "<br>";

//Chemin complet du fichier

echo "Chemin complet du fichier ". $_SERVER["SCRIPT_FILENAME"] ;

echo "<br>";

//Parametre de Requete

echo "Parametre de Requete ". $_SERVER["QUERY_STRING"] ;

echo "<br>";

//Chemin à partir du dossier racine sans le nom du fichier

echo "Chemin à partir du dossier racine sans le nom du fichier " .
$_SERVER["REQUEST_URI"] ;

echo "<br>";

//Chemin du fichier à partir du dossier racine

echo "Chemin du fichier à partir du dossier racine " .
$_SERVER["SCRIPT_NAME"] ;

echo "<br>";
```

CHAPITRE V

Fonctions et Fichiers

- Fonctions
 - Définition
 - Appel

Exercices Fonctions

Écrire les fonctions de validations suivantes:

- **bisextile()**: vérifie si une année est bisextile ou pas
- **nbre_jours_un_mois_dans_annee()**: retourne le nombre de jours d'un mois dans une année .
- **date_valide()**: vérifie si une date est valide ou pas
- **change_format()**: formate une date en français ou en anglais
- **date_suivante()** : retourne la date après un nombre de jour
- **date precedente()**: retourne la date qu'il faisait avant ce nombre de jours .
- **est_vide()** : qui vérifie si une valeur est vide ou pas
- **est_entier()**: qui vérifie si une valeur est un entier ou pas
- **est_reel()**: qui vérifie si une valeur est un entier ou pas
- **est_email()**: qui vérifie si une valeur est un email ou pas
- **est_premier()**: qui vérifie si un nombre est premier ou pas
- **est_une_phrase()**: qui vérifie si une chaîne est une phrase ou pas
- **enleve_espace_initiale()**: enlève les espaces inutiles d'une phrase
- **est_numero_tel()**: vérifie si un numero est valide ou pas
- **genrate_matricule()**: permet de générer le matricule suivant le format suivant LLL-LLL-LLL

CHAPITRE VI

Upload de Fichiers, Fichiers et Date

A) Upload de Fichiers (`$_FILES`)

- Fonctions
 - `file_exists()`
 - `move_uploaded_file()`
 - `pathinfo()`

Exemple :

a) Structure du Dossier

- `upload_files`
 - `upload`
 - `index.php`

b) code index.php

```
<?php

//5Mo maximum
define("MAX_UPLOAD_SIZE",5 * 1024 * 102);

// Vérifier si le formulaire a été soumis
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Vérifie si le fichier a été uploadé sans erreur.
    if(isset($_FILES["photo"]) && $_FILES["photo"]["error"] == 0){

        //Extensions Valides
        $allowed = [
            "jpg" => "image/jpg",
            "jpeg" => "image/jpeg",
            "gif" => "image/gif",
            "png" => "image/png"
        ];

        //Nom de la photo
        $filename = $_FILES["photo"]["name"];
        //type de la photo
        $filetype = $_FILES["photo"]["type"];
        //taille de la photo
        $filesize = $_FILES["photo"]["size"];

        // recupere l'extension du fichier
        $ext = pathinfo($filename, PATHINFO_EXTENSION);
        if(!array_key_exists($ext, $allowed)) die("Erreur : Veuillez
sélectionner un format de fichier valide.");

        // Vérifie la taille du fichier -
        if($filesize > MAX_UPLOAD_SIZE) die("Error: La taille du
fichier est supérieure à la limite autorisée.");

        // Vérifie le type MIME du fichier
        if(in_array($filetype, $allowed)){
            // Vérifie si le fichier existe avant de le télécharger.
            if(file_exists("upload/" . $_FILES["photo"]["name"])){
                echo $_FILES["photo"]["name"] . " existe déjà.";
            } else{
                move_uploaded_file($_FILES["photo"]["tmp_name"],
"upload/" . $_FILES["photo"]["name"]);
                echo "Votre fichier a été téléchargé avec succès.";
            }
        } else{
    }
}
```

```
        echo "Error: Il y a eu un problème de téléchargement de
votre fichier. Veuillez réessayer.";
    }
} else{
    echo "Error: " . $_FILES["photo"]["error"];
}
?>

<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Formulaire d'upload de fichiers</title>
</head>
<body>
    <form action="" method="post" enctype="multipart/form-data">
        <h2>Upload Fichier</h2>
        <label for="fileUpload">Fichier:</label>
        <input type="file" name="photo" id="fileUpload">
        <input type="submit" name="submit" value="Upload">
        <p><strong>Note:</strong> Seuls les formats .jpg, .jpeg, .jpeg,
.gif, .png sont autorisés jusqu'à une taille maximale de 5 Mo.</p>
    </form>
</body>
</html>
```

B) Fichiers Json

JSON (JavaScript Object Notation) est un format d'échange de données léger. Il est facile à lire et à écrire pour les humains ,et facile à analyser et à générer pour les machines . Il est basé sur un sous-ensemble du langage de programmation

JavaScript Standard ECMA-262 3e édition - décembre 1999.

JSON est un format de texte totalement indépendant du langage mais qui utilise des conventions familières aux programmeurs de la famille C de langages, y compris C, C ++, C #, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal.

- **Array to Json**
 - `json_encode()`
- **Json to Array**
 - `json_decode()`
- **Récupération des données d'un Fichier Json**
 - `file_get_contents()`
 - `file_exist()`
- **Ajout des données dans un Fichier Json**
 - `file_put_contents()`
- **Modification des données d'un Fichier Json**
 - `file_put_contents()`

Exemple :

a) Structure du Dossier

- `data.json`
- `index.php`

data.json

```
[  
    {  
        "id":1,  
        "reference":"Ref001",  
        "Description":"Lorem ipsum dolor sit amet consectetur",  
        "prix":500000,  
        "zone":  
            {  
                "id":1,  
                "nom":"zone1"  
            },  
        "type":"chambre"  
    },  
    {  
        "id":2,  
        "reference":"Ref002",  
        "Description":"Lorem ipsum dolor sit amet consectetur",  
        "prix":1500000,  
        "zone":  
            {  
                "id":1,  
                "nom":"zone2"  
            },  
        "type":"chambre"  
    }  
]
```

● index.php

```
function get_all_biens():array{  
    $file_content=file_get_contents(FILE_NAME);  
    $arr_biens=json_decode($file_content,true);  
    return $arr_biens;  
}  
  
function add_bien(array $bien):array{  
    $bien['id']=uniqid();  
}
```

```
$arr_biens=get_all_biens();
$arr_biens[]=$bien;
$file_input=json_encode($arr_biens);
file_put_contents(FILE_NAME,$file_input);
return $arr_biens;
}

var_dump(add_bien($bien=[

    "id"=>1,
    "reference"=>"Ref001",
    "Description"=>"Lorem ipsum dolor sit amet consectetur adipisicing elit. At commodi quia quis. Odit itaque dicta distinctio cumque placeat possimus iure, amet fugiat rem at voluptas eligendi saepe accusantium neque doloribus?",
    "prix"=>500000,
    "zone"=>[
        "id"=> 1 ,
        "nom"=> "zone 1"
    ],
    "type"=>"chambre"
]));
```

CHAPITRE VI

POO et -MVC en

PHP

I. Notion POO(Classe, Interface,Héritage)

a) Classe et Objet

La classe renferme l'ensemble des propriétés et de méthodes qui servent à définir l'identité de l'objet qui en découle (l'instance de classe).

1. Syntaxe de Déclaration

Structure d'une classe PHP

```
<?php
    class NomClass{
        //attributs
        visibilité $attribut1;
        visibilité $attribut2;
        ...
        visibilité $attributn;
        //méthodes
    }
?>
```

visibilité : public, protected ou private

2. Crédation d'un Objet

Création d'un objet : instantiation (dans index.php)

```
$nomObjet = new NomClasse();
```

Explication

- pour créer un objet on utilise l'opérateur `new` et on fait appel au constructeur
- il existe un constructeur par défaut pour chaque classe en **PHP**
- il est toujours possible de créer son propre constructeur
- la création d'un constructeur écrasera le constructeur par défaut

3. Remarque

Pour attribuer des valeurs aux attributs, on utilise `$this` et `->`

- `$this` : désigne l'objet courant
- `->` : pointer un élément d'un objet (méthodes ou attributs)

Remarques

- Le mot-clé `self` ⇒ à utiliser dans la classe pour accéder à élément `static`
- `::` (Resolution Operator) ⇒ à utiliser avec `self` (comme `$this` et `->`)

this vs self

- on utilise `$` avant `this` mais pas avant l'attribut qui le suit
- on utilise `$` avant l'attribut de la classe qui suit `self` mais pas avant `self`

4. Exemple de Classe

```
chrome://bookmarks  
class Personne  
{  
    public int $num;  
    public string $nom;  
    public string $prenom;  
  
    public function __construct()  
    {}  
  
    function __destruct()  
    {}  
  
    public function __toString(): string  
    {  
        return $this->num . " " . $this->nom . " " . $this->prenom;  
    }  
}
```

5. Encapsulation

Démarche

- 1 Bloquer l'accès directe aux attributs (mettre la visibilité à `private`)
- 2 Définir des méthodes publiques qui contrôlent l'affectation de valeurs aux attributs (les `setter`)

Convention

- Utiliser la visibilité `private` ou `protected` pour les attributs
- Utiliser la visibilité `public` pour les méthodes

6. Déclaration et Manipulation des Objets

```
$personne = new Personne();  
$personne->setNum(100);  
$personne->setNom("wick");  
$personne->setPrenom("john");  
echo $personne;  
/* affiche 100 wick john */
```

Testons aussi avec une valeur négative pour num

```
$personne = new Personne();  
$personne->setNum(-100);  
$personne->setNom("wick");  
$personne->setPrenom("john");  
echo $personne;  
/* affiche 0 wick john */
```

7. Attributs Constantes de Classe

Constante de classe

C'est comme un attribut statique constant

Syntaxe

```
<?php
    class NomClasse{
        //attributs
        ...
        //constantes
        const NOM_CONSTANTE = 'value';
        //méthodes
        ...
    }
?>
//Pour accéder :
NomClasse::NOM_CONSTANTE
```

8. Synthèse

Récapitulatif

- On recommande d'avoir une seule classe par fichier
- Une classe peut avoir comme attribut un objet d'une deuxième classe (définie dans un deuxième fichier)
- Il faut inclure la deuxième classe (avec `include` ou `require`) pour l'utiliser
- Et si dans un fichier on utilise plusieurs classes, Faut-il les inclure toutes une par une ?
- Non on peut utiliser la fonction `spl_autoload_register` pour faire l'auto-chargement

b) Classe et Objet

i) Bidirectionnel/Unidirectionnel

1) OneToOne

2) ManyToOne ou OneToMany

3) ManyToMany

ii) Héritage

1) Définition

L'héritage, quand ?

- Lorsque deux ou plusieurs classes partagent plusieurs attributs (et méthodes)
- Lorsqu'une Classe1 est (**une sorte de**) Classe2

Forme générale

```
class ClasseFille extends ClasseMère
{
    // code
};
```

Exemple

Classe Mere Personne

```
chrome://bookmarks

class Personne
{
    public int $num;
    public string $nom;
    public string $prenom;

    public function __construct()
    {}

    function __destruct()
    {}

    public function __toString(): string
    {
        return $this->num . " " . $this->nom . " " . $this->prenom;
    }
}
```

Classe Fille Enseignant

Contenu de la classe Enseignant

```
class Enseignant extends Personne
{
    public function __construct(int $num, string $nom, string $prenom)
    {
        parent::__construct($num, $nom, $prenom);
    }
    function __destruct()
    {}
}
```

Contenu de la classe Etudiant

```
class Etudiant extends Personne
{
    public function __construct(int $num, string $nom, string $prenom)
    {
        parent::__construct($num, $nom, $prenom);
    }
    function __destruct()
    {}
}
```

Classe Fille Etudiant

À partir de la classe Enseignant

- On ne peut avoir accès direct à un attribut de la classe mère
- C'est-à-dire, on ne peut faire `$this->_num` car les attributs ont une visibilité `private`
- Pour modifier la valeur d'un attribut privé de la classe mère, il faut
 - soit utiliser les getters/setters
 - soit mettre la visibilité des attributs de la classe mère à `protected`

2) Classe Abstraites

(a) Définition et Exemple

Classe abstraite

- C'est une classe qu'on ne peut instancier
- On la déclare avec le mot-clé `abstract`

Syntaxe

```
abstract class NomClasse
{
    // le code
}
```

Exemple

```
abstract class Personne{
    // attributs
    ...
    // méthodes
    ...
}

$Pers = new Personne();
//ça génère une erreur fatale
```

(b) Méthodes Abstraites

Méthode abstraite

- C'est une méthode non implémentée (sans code ⇒ sans accolades). Nous devons juste préciser :
 - sa visibilité (public, protected ou private).
 - sa signature (nom et type de paramètres) [si elle prend de paramètres].
- Une méthode abstraite doit être déclarée dans une classe abstraite
- Une méthode abstraite doit être implémentée par les classes filles de la classe abstraite

Syntaxe

```
abstract public function nomMethode();
```

(c) Exemple

Déclarons une méthode abstraite afficherCaracteristiques() dans Personne

```
abstract public function afficherCaracteristiques(): void;
```

Remarque

- La méthode afficherCaracteristiques() dans Personne est soulignée en rouge car la classe doit être déclarée abstraite
- Placer le curseur sur la méthode afficherCaracteristiques() et cliquez sur Make type 'Personne' abstract

NB :

- Une méthode doit se trouver dans une classe abstraite ou une dans une interface.
- Une Classe abstraite peut définir des méthodes abstraites ou concrètes.

(d) Héritage avec Classe Mère Abstraite

Lorsque classe hérite d'une classe abstraite elle doit soit:

- Redéfinir toutes les méthodes abstraites en des méthodes concrètes
- de se transformer en une classe abstraite

3) Classe Finale

(a) Définition et Exemple

Classe finale

C'est une classe qui ne peut avoir de classes filles

Syntaxe

```
final class NomClasse{  
    // le code  
}
```

(b) Exemple

Exemple

```
final class Personne{  
    // le code  
}  
  
class Etudiant extends Personne{  
    // le code  
}  
  
//ça génère une erreur fatale
```

(c) Méthodes Final

Méthode finale

C'est une méthode qu'on ne peut redéfinir

```

class Personne
{
    // code précédent
    final public function decire(){
        ...
    }
}
class Etudiant extends Personne
{
    // code précédent
    final public function decire(){
        ...
    }
}

```

Le code précédent génère une erreur fatale :

Cannot override the final method from Personne

4) Résumé

Remarques

- Une classe abstraite ne doit pas forcément contenir une méthode abstraite
- Une classe finale ne doit pas forcément contenir une méthode finale
- Une méthode finale ne doit pas forcément être dans une classe finale

5) Quelques Fonctions utiles

Fonctions utiles pour les classes

- `get_class($obj)` : retourne le nom de la classe de l'objet \$obj
- `get_class_methods(NomClasse)` : retourne un tableau contenant les méthodes de la classe NomClasse
- `class_exists(NomClasse)` : vérifie si la classe NomClasse existe
- `get_object_vars($obj)` : retourne un tableau contenant les attributs de l'objet (\$obj)
- `get_parent_class($obj)` : Retourne le nom de la classe mère d'un objet \$obj
- **autres** : `is_a`, `is_subclass_of`, `method_exists`, `property_exists`...

c) Interface

1) Définition

Une interface

- déclarée avec le mot-clé `interface`
- comme une classe abstraite (impossible de l'instancier)
 - dont toutes les méthodes sont abstraites
 - qui n'a pas d'attribut
- un protocole, un contrat : toute classe qui hérite d'une interface doit implémenter toutes ses méthodes
- Pas besoin du mot-clé `abstract` pour les méthodes d'une interface

Syntaxe

```
interface NomInterface
{
    ...
}
```

2) Exemple

Définissons la signature de ces deux méthodes dans l'interface IMiseEnForme

```
interface IMiseEnForme
{
    public function afficherNomMajuscule(): void;
    public function afficherPrenomMajuscule(): void;
}
```

3) Relation Implementation

Pour hériter d'une interface, on utilise le mot-clé `implements`

```
class Personne implements IMiseEnForme
{
    ...
}
```

La classe `Personne` est soulignée en rouge

- Placer le curseur sur la classe `Personne`
- Dans le menu affiché, sélectionner `Add unimplemented methods`

4) Instantiation

Pour tester, voici le contenu d'`index.php`

```
$etudiant = new Etudiant(101, "maggio", "carol", 'master');

$etudiant->afficherNomMajuscule();
// affiche MAGGIO

$etudiant->afficherPrenomMajuscule();
// affiche CAROL
```

5) Remarques

Lorsque classe implémente une interface, elle doit soit:

- Redéfinir toutes les méthodes abstraites en des méthodes concrètes
- de se transformer en une classe abstraite

Remarques

- Une interface peut hériter de plusieurs autres interfaces (mais pas d'une classe)
- Pour cela, il faut utiliser le mot-clé `extends` et pas `implements` car une interface n'implémente jamais de méthodes.

Exemple

```
interface I extends I2, I3
    ...
}
```

Fonctions utiles pour les interfaces

- `get_declared_interfaces()` : retourne un tableau contenant toutes les interfaces déclarées
- `class_implements()` : retourne les interfaces implémentées par une classe ou une interface donnée
- `class_exists()` : vérifie si une classe a été définie
- `interface_exists` : vérifie si une interface existe

NB:

- L'héritage multiple n'est pas autorisé en **PHP**.
- Mais une classe peut implémenter plusieurs interfaces.
- Les interfaces ne prennent pas d'attributs.
- Nous ne pouvons pas implémenter deux interfaces qui ont deux méthodes identiques.

d) Trait

Un trait

- Un moyen de réutiliser le code (les méthodes)
- Déclaré avec le mot-clé `trait`

Notation

- Une classe (fille) étend (`extends`) une classe (mère)
- Une classe implémente (`implements`) une interface
- Une classe utilise (`use`) un trait

Syntaxe

```
trait NomTrait
{
    ...
}
```

Exemple:

```
trait France
{
}
```

Ajoutons une méthode à ce trait

```
trait France
{
    public function salutation()
    {
        echo "salut";
    }
}
```

Utilisons le trait France dans la classe Personne

```
class Personne implements IMiseEnForme
{
    // attributs, constantes...

    use France;

    // méthodes

}
```

Pour tester le trait dans index.php

```
$etudiant = new Etudiant(101, "maggio", "carol", 'master');
$etudiant->salutation();
// affiche salut
```

Propriétés

- Une classe peut utiliser un ou plusieurs traits.
- Une classe peut utiliser plusieurs traits qui ont une méthode avec le même nom.
- Il suffit de préciser quelle méthode sera utilisée par les objets de la classe.

Définissons un deuxième trait America

```
trait America
{
    public function salutation()
    {
        echo "hi";
    }
}
```

Si la classe Personne utilise les deux traits, elle doit préciser quelle méthode salutation préfère utiliser

```
class Personne implements IMiseEnForme
{
    // attributs, constantes...

    use France, America {
        America::salutation insteadof France;
    }

    // méthodes

}
```

Pour tester le trait dans index.php

```
$etudiant = new Etudiant(101, "maggio", "carol", 'master'
);
$etudiant->salutation();
// affiche hi
```

e) Namespace

i) Définition

Namespace ?

- Un concept repris du langage C++
- Un espace de stockage abstrait (n'existe pas physiquement)
- Un répertoire virtuel
- Conçu pour développer des structures fonctionnelles (variable, constante, fonction...) sans avoir de conflit de nom

ii) Exemple

Considérons le fichier `salutationFr.php` défini dans un répertoire `fonctions`

```
function direBonjour(): void
{
    echo "Bonjour";
}
```

Et un deuxième fichier `salutationEn.php`

```
function direBonjour(): void
{
    echo "Good morning";
}
```

iii) Problématique:

Dans `index.php`, commençons par inclure les deux fichiers

```
<?php
    include 'fonctions/salutationEn.php',
    include 'fonctions/salutationFr.php';
?>
```

Dans le body d'`index.php`, appelons la fonction `direBonjour()`

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

En exécutant, le résultat est

Fatal error: Cannot redeclare `direBonjour()`

iv) Solution:

Première solution

Faire include d'un seul fichier

Mais

dans certains cas on a besoin d'utiliser les deux



Deuxième solution

Définir chacune de ses fonctions dans un namespace différent.

Solution avec les namespaces

PHP

Bookmarks
chrome://bookmarks

Définissons un namespace French dans salutationFr.php

```
namespace French;

function direBonjour(): void
{
    echo "Bonjour";
}
```

Et un deuxième English dans salutationEn.php

```
namespace English;

function direBonjour(): void
{
    echo "Good morning";
}
```

Il ne reste qu'à préciser le namespace et l'appel

```
<?php
    French\direBonjour();
    English\direBonjour();
?>
```

v) Alias des Namespace

Dans index.php, on peut donner des alias à nos namespaces avec use ... as

```
<?php
    include 'fonctions/salutationEn.php';
    use English as en;
    include 'fonctions/salutationFr.php';
    use French as fr;
?>
```

Il reste qu'à utiliser les alias au moment de l'appel

```
<?php
    fr\direBonjour();
    en\direBonjour();
?>
```

vi) Imbrication de NameSpace

On peut aussi imbriquer les namespaces (contenu de salutationEn.php)

```
namespace English;

function direBonjour(): void
{
    echo "Good morning";
}
namespace English\American;

function direBonjour(): void
{
    echo "Good mornin";
}
```

vii) Remarques

Dans index.php, on commence par les include

```
<?php
    include 'fonctions/salutationEn.php';
    use English as en;
    use English\American as am;
    include 'fonctions/salutationFr.php';
    use French as fr;
?>
```

Il reste qu'à utiliser les alias au moment de l'appel

```
<?php
    fr\direBonjour();
    en\direBonjour();
    am\direBonjour();
?>
```

Propriétés

- On peut déclarer plusieurs namespaces par fichier (mais déconseillé)
- Par défaut (si on ne déclare aucun namespace), on est dans le namespace global
- Pour appeler une fonction du namespace global, on utilise \
- Pour retourner au namespace global, il suffit d'écrire namespace
- On peut déclarer plusieurs sous-namespaces, comme pour les dossiers sous windows, il faut les séparer par
Espace1\Espace2\Espace3
Si les namespaces Espace1 et Espace2 n'existent pas, ils seront créés.
- La constante __NAMESPACE__ contient toujours le nom du namespace courant
- Pour accéder à un élément d'un namespace différent : namespace\nomElement

Le namespace doit être le même que le dossier dans lequel se trouve la classe.

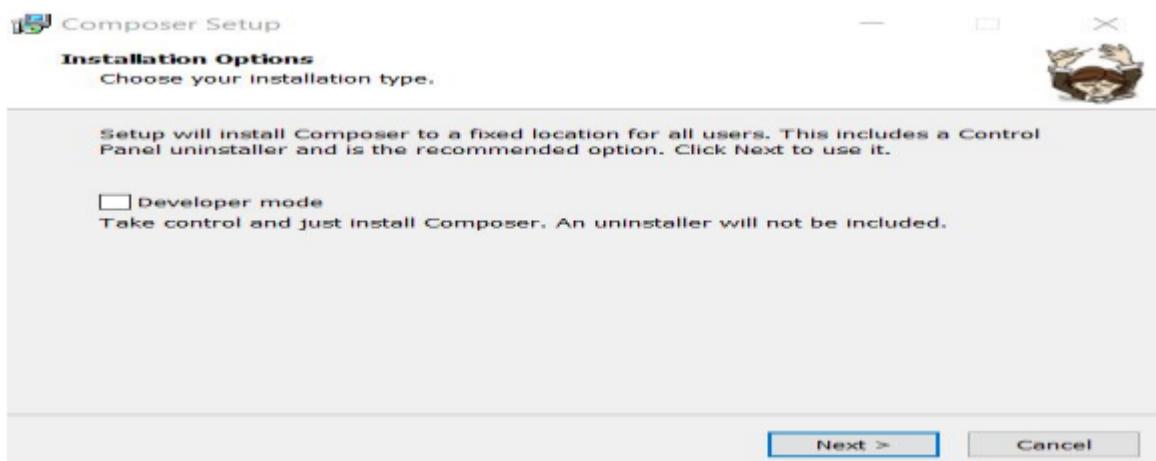
II. Gestionnaire de Dépendance : Composer

Composer est un gestionnaire de dépendances pour PHP. C'est un outil simple et fiable que les développeurs utilisent pour gérer et intégrer des paquets ou des bibliothèques externes dans leurs projets basés sur PHP. Ainsi, ils n'ont pas à créer leurs pages ou applications web à partir de zéro.

1. Installation de composer a) Windows

Nous vous recommandons d'utiliser XAMPP ou WAMP à cette fin, car le processus est simple et vous pouvez le terminer en quelques minutes. Une fois que XAMPP ou WAMP est installé, téléchargez la dernière version de Composer.

Lancez l'assistant d'installation de Composer. Lorsqu'il vous demande d'activer le mode développeur, ignorez-le et poursuivez le processus d'installation.



2. Norme PSR-4: Autoloader

- a) **Configurer Composer**
`composer init`
- b) **Configuration du autoload**

- **Dans le fichier `composer.json`**

```
"""autoload": {
    "psr-4": {
        "App\\Controllers\\": "controllers/",
        "App\\Models\\": "models/",
        "App\\Core\\": "core/"
    }
},
```

NB : A chaque fois qu'on change le fichier composer.json, il faudra faire un *composer update*.

- *Dans le fichier index.php*

```
require_once "../vendor/autoload.php";
```

III. Notion MVC

MVC : c'est quoi ?

C'est un design pattern

- En français : Modèle-Vue-Contrôleur
- En anglais : Model-View-Controller

Historique

- introduit par Trygve Reenskaug en 1978
- utilisé pour la première fois par **Smalltalk** en 1980

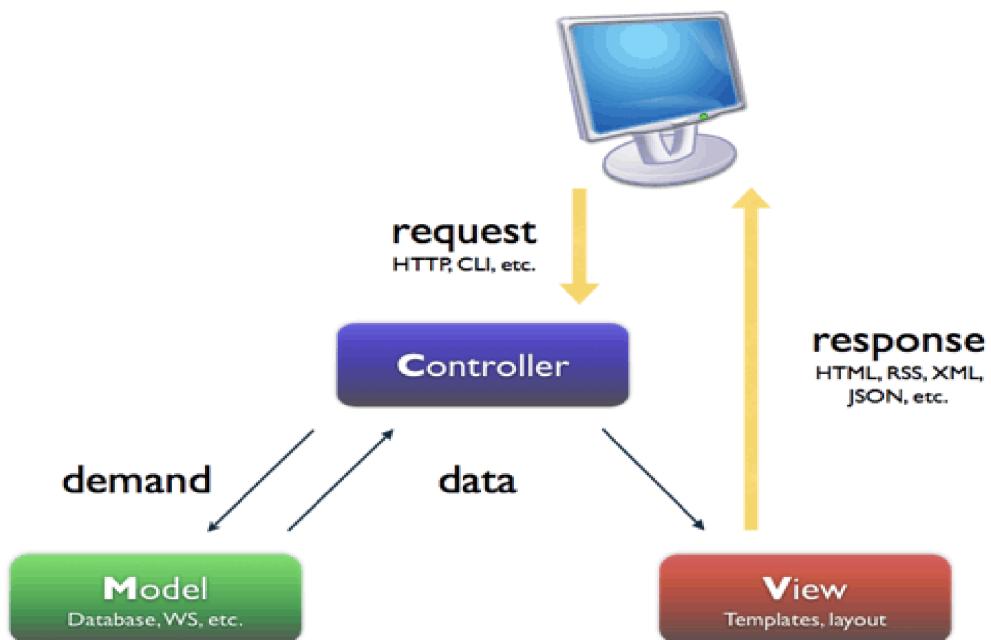
Principe

- permet de bien organiser son code source
- permet de savoir quel rôle joue précisément chaque fichier dans l'application
- consiste à séparer la logique du code en trois parties, modèle (modèle de données), vue (présentation, interface utilisateur) et contrôleur (logique de contrôle, gestion des événements, synchronisation...)

C'est-à-dire

- Modèle : la partie qui concerne les données (base de données, fichiers...) de notre application (site...). Il dispose de toutes les méthodes d'accès aux données (insert, delete, update, select)
- Vue : la partie qui concerne l'affichage : l'interface avec laquelle l'utilisateur interagit (page HTML...). C'est la vue qui s'occupe de la visualisation des données retournées par le modèle
- Contrôleur : c'est l'intermédiaire entre le modèle et la vue. Il demande les données au modèle, les analyse, et prend ensuite des décisions et renvoie le texte à afficher à la vue.

MVC : schématisation



MVC : explication

Déroulement

- ① Le client envoie une requête depuis la vue (son navigateur)
- ② Le contrôleur intercepte et analyse la requête du client
- ③ Le contrôleur détermine quelle partie du modèle est concernée afin d'effectuer les traitements nécessaires
- ④ Le modèle s'occupe de l'interaction avec les données, applique les règles métier et renvoie les données (dénudés de toute présentation) au contrôleur
- ⑤ Le contrôleur sélectionne la vue correspondante et lui injecte les données
- ⑥ La vue présente les données au client

Et le MVC 2 ?

Exemple

- Le MVC est un peu difficile à mettre en place à cause de la multitude de contrôleurs
- Donc une nouvelle version a été introduite : MVC 2
- Il s'agit du même modèle avec un seul contrôleur pour orienter les requêtes

Frameworks basés sur l'architecture MVC (2)

Exemple

- 1 CakePHP
- 2 FuelPHP
- 3 Jelix
- 4 Laravel
- 5 Symfony
- 6 Zend
- 7 ...

CHAPITRE VII

PDO

A. Définition

C'est une extension définissant l'interface pour accéder à une base de données avec PHP

B. Les méthodes Utiles

a. Gestionnaire de connexion PDO aux SGBD MYSQL

```
$pdo = new PDO('mysql:dbname=nom_bd;host=@server_bd',
    'user_bd', 'password');
```

b. Modifier un Attribut a PDO:

```
$pdo->setAttribute (Attribut, Valeur);
```

c. Les Attributs De PDO

i. Gestion des Erreurs

PDO::ATTR_ERRMODE :Attribut qui gère le rapport d'erreurs et a comme valeurs :

- **PDO::ERRMODE_SILENT** (Par Défaut): assigne simplement les codes d'erreur.
- **PDO::ERRMODE_WARNING**: émet une alerte E_WARNING.
- **PDO::ERRMODE_EXCEPTION** : émet un exception.

ii. Mode de Récupération des Données

PDO::ATTR_DEFAULT_FETCH_MODE: Gère le Mode De Récupération des Données et a comme valeurs:

- **PDO::FETCH_ASSOC**: retourne un tableau indexé par le nom de la colonne comme retourné dans le jeu de résultats .
- **PDO::FETCH_NUM** : retourne un tableau indexé par le numéro de la colonne comme elle est retourné dans votre jeu de résultat, commençant à 0.
- **PDO::FETCH_BOTH (défaut)**: retourne un tableau indexé par les noms de colonnes et aussi par les numéros de colonnes, commençant à l'index 0,comme retournés dans le jeu de résultats.
- **PDO::FETCH_OBJ**: retourne les Valeurs sous la forme d'un tableau d'Objets

d. Ecriture et Exécution des Requêtes

i. Requête Non Préparée ou Non Paramétrée:

Une requête est non préparée lorsque les variables de la requête sont injectées à la définition de la requête.

Pour exécuter les requêtes non préparées , on utilise :

```
$pdo->query ($requete);
```

Exemple :

```
$pdo->query ("select * from user where id_user=$user");
```

ii. Requête Préparée ou Paramétrée :

Une requête est préparée lorsque les variables de la requête sont injectées lors de l'exécution. A la définition de la requête on remplace les variable par le joker ?.

Pour exécuter les requêtes non préparées , on utilise :

\$req=\$pdo->prepare (\$sql);

Exemple :

\$req=\$pdo->prepare("select * from user where id_user=?");

\$req->execute (array (1));

iii. Recuperation des Donnees

- **Un Enregistrement \$variables= \$req->fetch();**
- **Plusieurs Enregistrement \$variables= \$req->fetchAll();**

NB :

\$pdo->lastInsertId : Retourne l'identifiant de la dernière ligne insérée ou la valeur d'une séquence

\$req->rowCount():Retourne le nombre de lignes de la requête

CHAPITRE VIII

Réalisation du Projet: Gestion des Inscriptions

I. Description du Projet

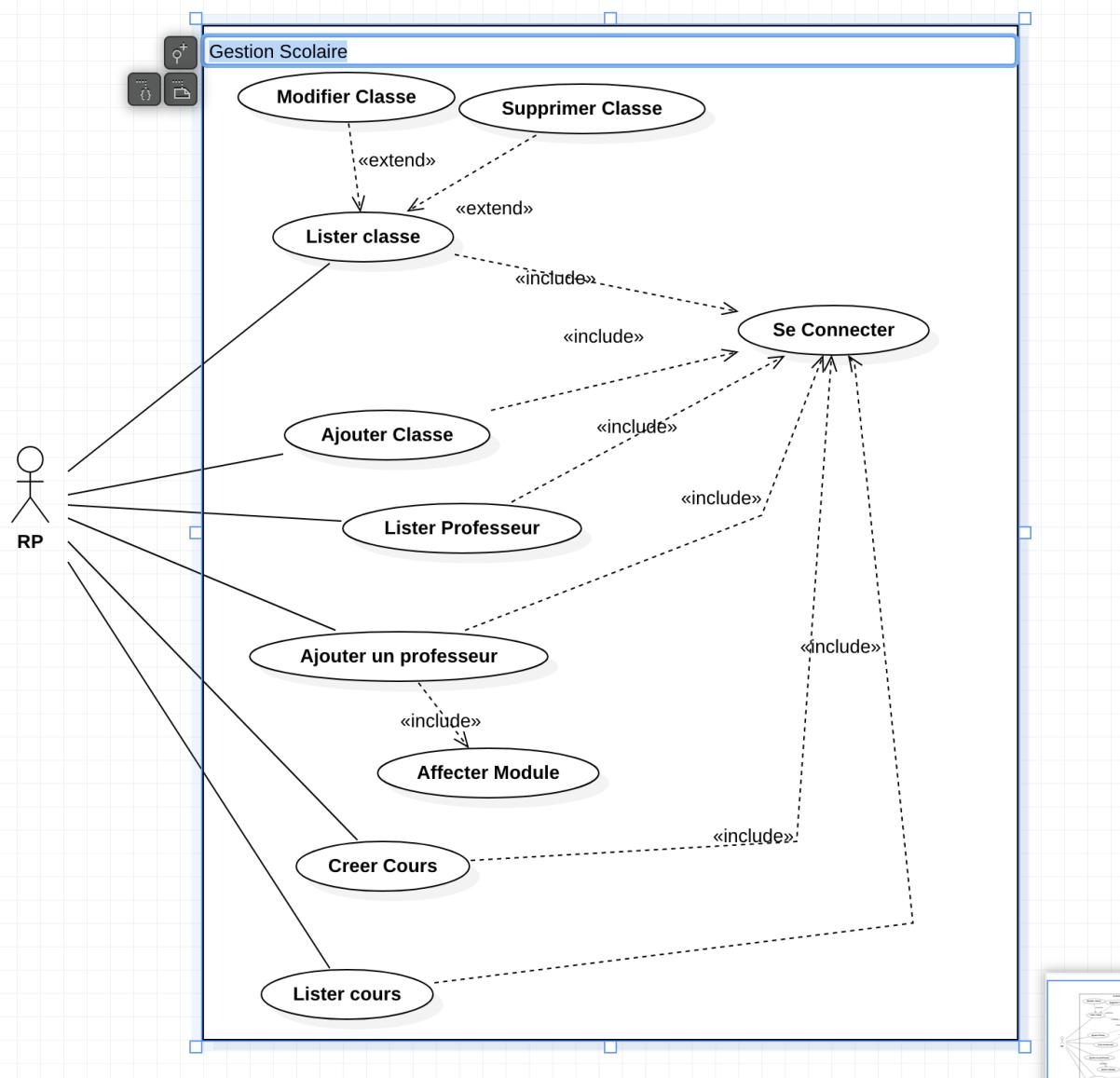
Mettre en place un système de gestion des inscriptions :

- Le Responsable Pédagogique (RP):
 - Lister,ajouter,modifier et supprimer des classes
 - Lister,ajouter des professeurs ou des modules
 - Lister, Créer des cours
 - connexion
- L' Attaché de Classe (AC) peut :
 - inscrire ou réinscrire des étudiants dans une année,lors de l'inscription on crée un dossier pour l'étudiant
 - Lister les Étudiants inscrits dans une année et filtrer par classe
 - Lister les cours d'une classe et filtrer par date et afficher les étudiants de ce cours
- Un étudiant (E) peut :
 - Lister ses cours et filtrer par date
- Un Professeur (E) peut :
 - Lister les cours d'une classe et filtrer par date et afficher les étudiants de ce cours

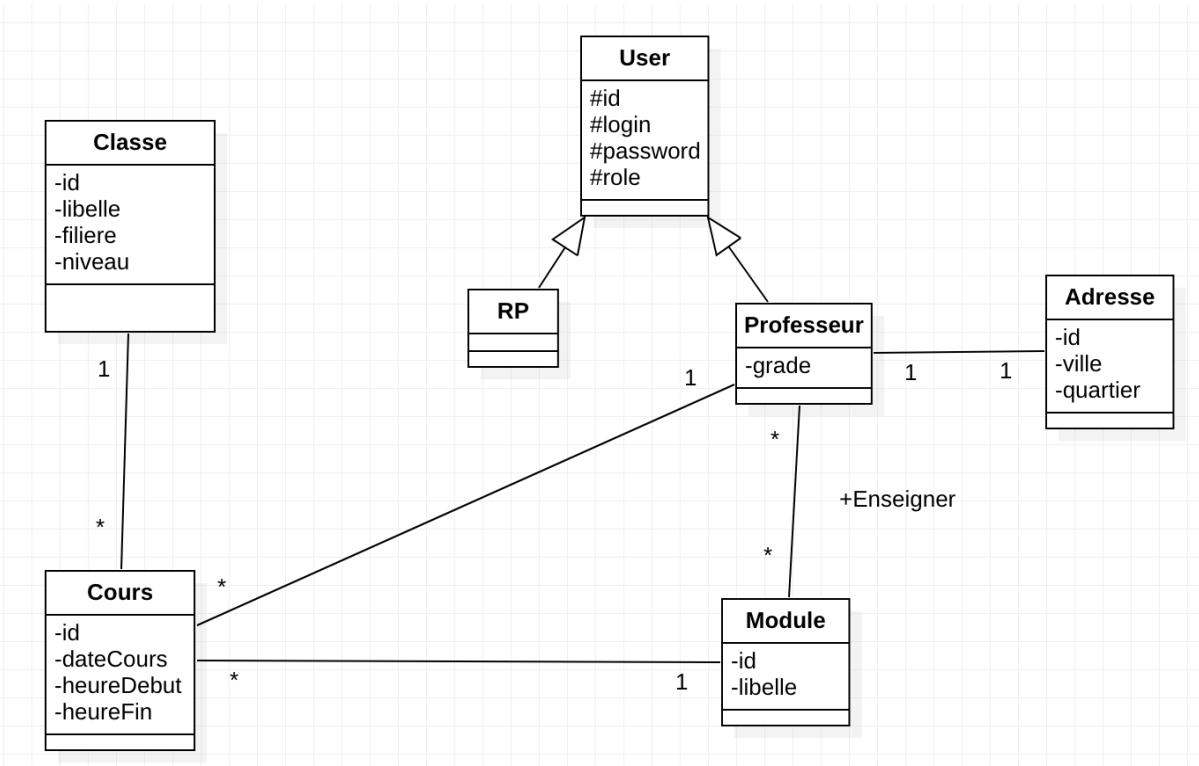
NB : Toutes les fonctionnalités sont accessibles après connexion

TAF

1. Diagramme de Use Case



2. Diagramme de Classe



3. Faire le MLD

user(id,login,password,grade,ville,quartier,role)
 classe(id,libelle,filiere,niveau)
 module(id,libelle)
 professeur_module(#module_id,#professeur_id)
 cours(id,date_cours,heure_debut,heure_fin,#classe_id,module_id,#professeur_id)

4. Créer la Base de Données

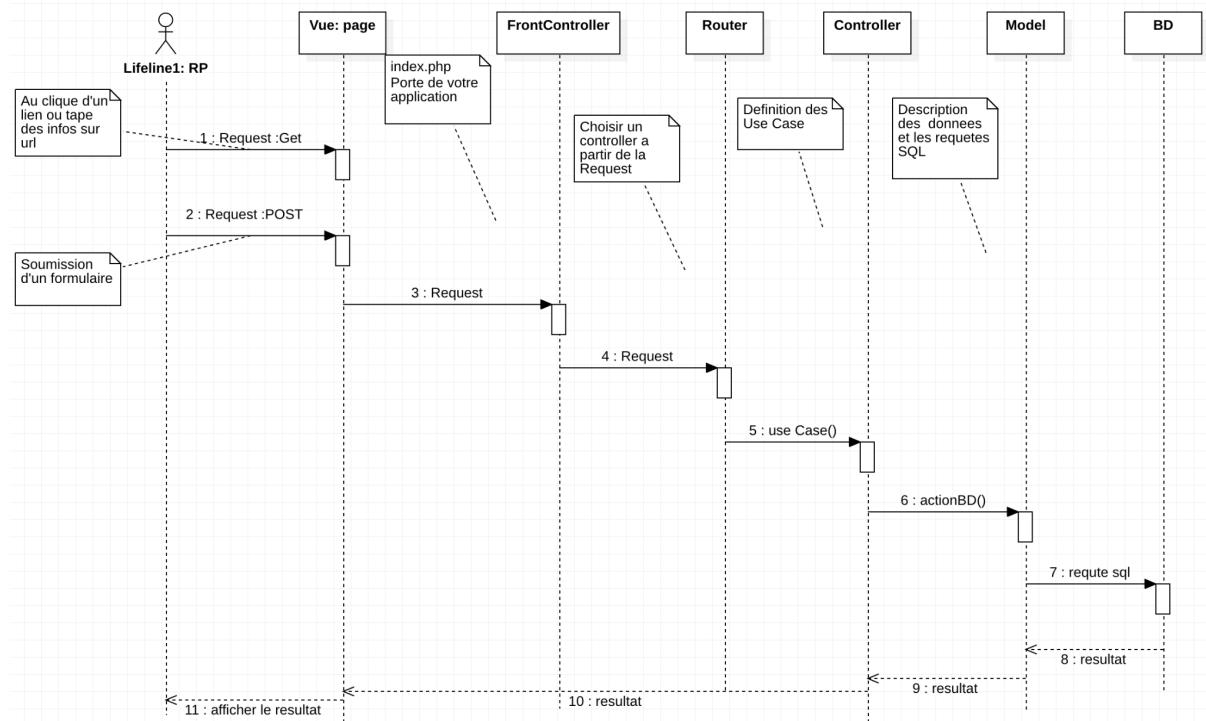
a. gestion_scolaire_I2

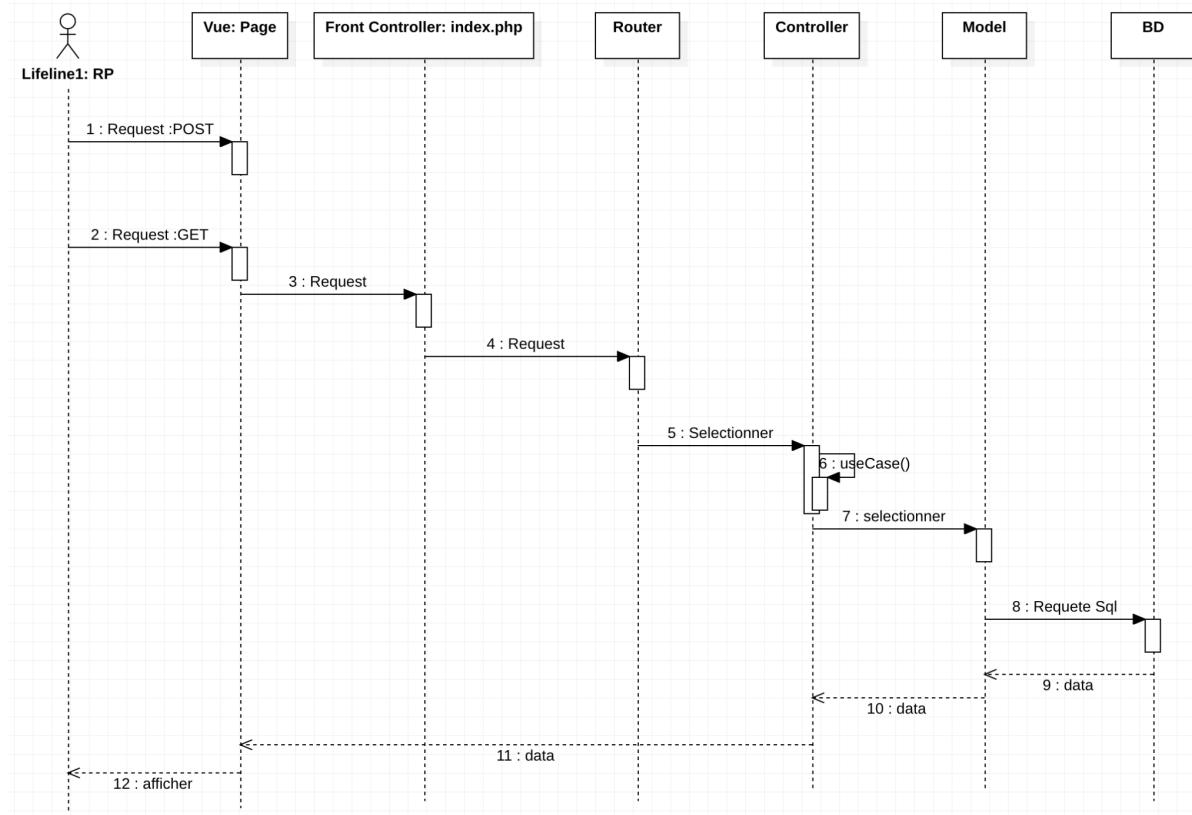
5. Architecture MVC(Model View Controller)

a. Lancement du serveur

php -S localhost:8000 -t public

b. Diagramme de sequence architecture MVC





6. Lancement de l'application

`php -S localhost:8000 -t public`

7. Component Model

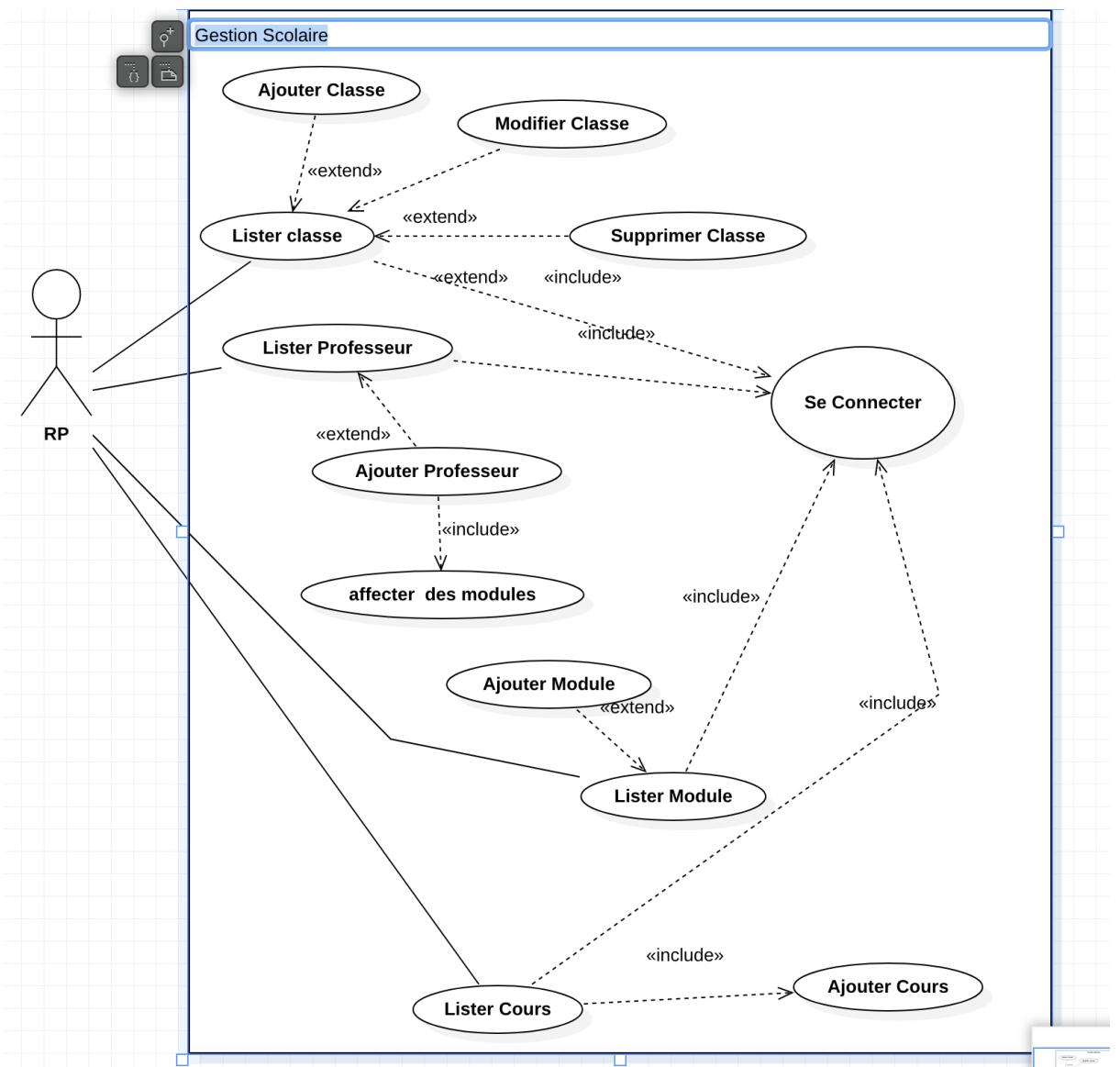
Dans un Model, on aura :

- Description des données => Diagramme de Classe
- Persistance des données => communication avec la BD(requêtes sql)

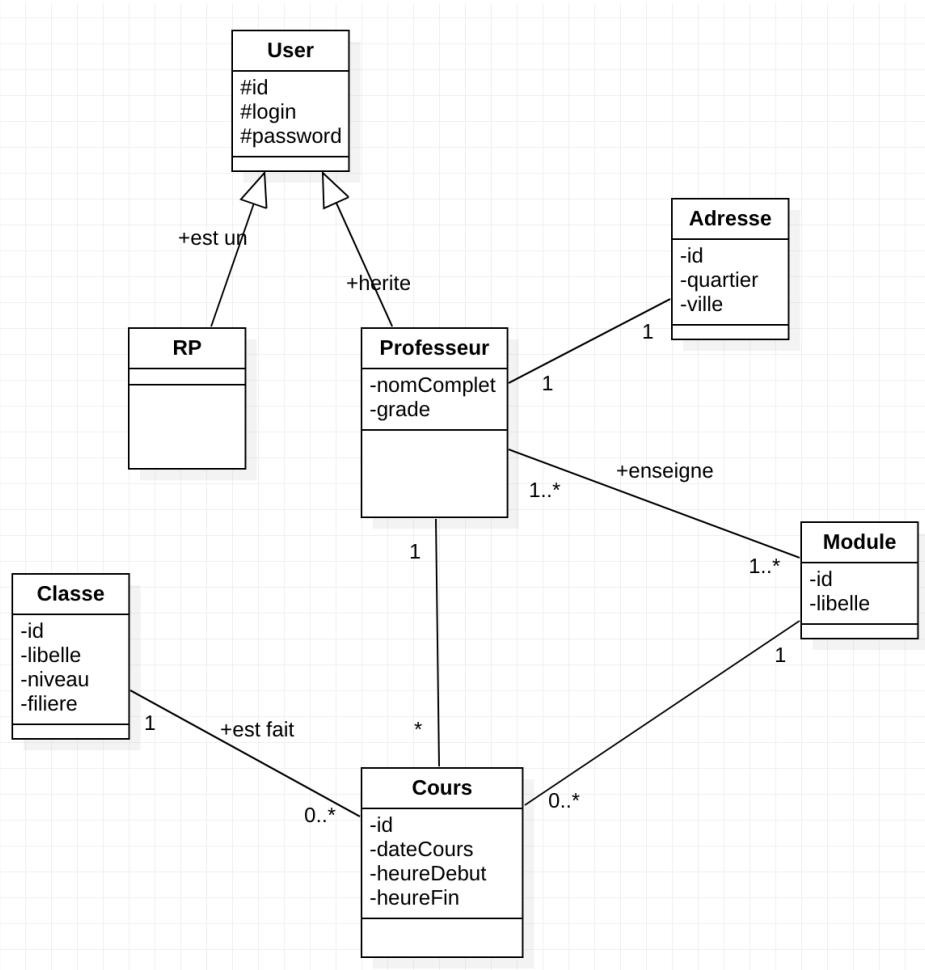
Mettre en place un système de gestion des inscriptions :

- **Le Responsable Pédagogique (RP):**
 - Lister,ajouter,modifier et supprimer des classes
 - Lister,ajouter des professeurs ou des modules
 - Lister, Créer des cours
 - connexion

1. Diagramme Use



2. Diagramme Classe



3. MLD

a. Règles

i. Heritage

Single Table : Toutes les classes qui participent dans la hiérarchie d'héritage seront transformées en une table.

NB : Lorsqu'on utilise l'approche `single Table`, il faudra ajouter un nouvel attribut qui indiquera le type utilisateur.

ii. OneToMany ou ManyToOne

Les classes qui participent dans cette relation deviennent des tables et l'association est traduite par un clé étrangère. L'identifiant de classe dont la multiplicité maximale la plus faible migre dans la classe dont la multiplicité maximale est la plus forte.

iii. ManyToMany

Les classes qui participent dans cette relation deviennent des tables et l'association est traduite en une table qui contient les identifiants des classes qui participent dans l'association.

iv. OneToOne

- **1..1 -> 1..1 :** fusionnés les deux classes en une table
- **0..1 -> 1..1 :** Les classes qui participent dans cette relation deviennent des tables et l'association est traduite par un clé étrangère. L'identifiant de classe dont la multiplicité minimale la plus faible migre dans la classe dont la multiplicité minimale est la plus forte.
- **0..1 -> 0..1 :** Les classes qui participent dans cette relation deviennent des tables et l'association est traduite en une table qui contient les identifiants des classes qui participent dans l'association.

b. Transformation

- user(id,login,password,nom_complet,grade,role,quartier,ville)
- classe(id,libelle,niveau,filiere)
- cours(id,date_cours,heure_debut,heure_fin,#classe_id,#professeur_id,#module_id)
- module(id,libelle)
- professeur_module(#module_id,#professeur_id)

4. BD

a. nom BD : gestion_scolaire_I2

5. Architecture MVC

a. Models

II. Création de la structure projet

- poo**

- core :** contient les composant de base de notre projet et réutilisable

- **models:** contient les composants d'accès aux données.
- **controllers:** contient les composants qui traitent les requêtes GET ou POST.
- **views:** représente les interface

III. Réalisation des composants Models

A. Diagramme de Classe Conception

B. Réalisation des Classes

1. Définir les classes du Diagramme de classe
 - a) Définition des attributs
 - b) Définition des getters et setters
 - c) Définition des fonctions acces aux donnees
 - (1) **all(): méthodes statiques**
 - (2) **find(): méthodes statiques**
 - (3) **where(): méthodes statiques**
 - (4) **orderBy(): méthodes statiques**
 - (5) **insert(): méthodes instances**
 - (6) **update(): méthodes instances**
2. Réalisation de l'interface **IModel**
3. Réalisation de classe abstraite **Model**
implemente **IModel**

