
Module 133

Version 1.0.0

ETML

avr. 18, 2020

Table des matières

No Module 133

Titre Réaliser des applications Web en Session-Handling.

Compétence Réaliser et tester une application Web selon cahier des charges avec un langage de programmation.

Objectifs opérationnels

1. Analyser la donnée, projeter la fonctionnalité et déterminer le concept de la réalisation.
2. Réaliser une fonctionnalité spécifique d'une application Web par Session-Handling, authentification et vérification de formulaire.
3. Programmer une application Web à l'aide d'un langage de programmation compte tenu des exigences liées à la sécurité.
4. Vérifier la fonctionnalité et la sécurité de l'application Web à l'aide du plan tests, verbaliser les résultats et, le cas échéant, corriger les erreurs.

Author ETML

Mise à jour avr. 18, 2020

Références

- Identification du module
- DEP

Table des matières

1.1 Définition et histoire

PHP (PHP Hypertext Preprocessor) est un langage de scripts permettant de réaliser des applications Web. Il est utilisé en parallèle avec du HTML (pour la mise en page), du CSS (pour la mise en forme) et au JavaScript (pour des animations, effets, vérifications, etc.).

1.1.1 Principales caractéristiques du langage PHP

- Langage interprété (pas compilé)
- Conçu pour le Web
- Open Source
- Grosse communauté, beaucoup de fonctionnalités
- Gratuit
- Rapidité, stabilité, sécurité
- Ecosystème très riche (nombreux frameworks, CMS, bibliothèques, etc.)
- Depuis la version 5, il peut être utilisé de manière orienté objet

Le PHP tourne sur des serveurs de type Linux. Il est possible de coder sur toutes les plateformes actuelles (Windows, Linux et Mac) en utilisant un serveur Web local. Pour Windows, le serveur utilisé sera uWamp : [uWamp](#).

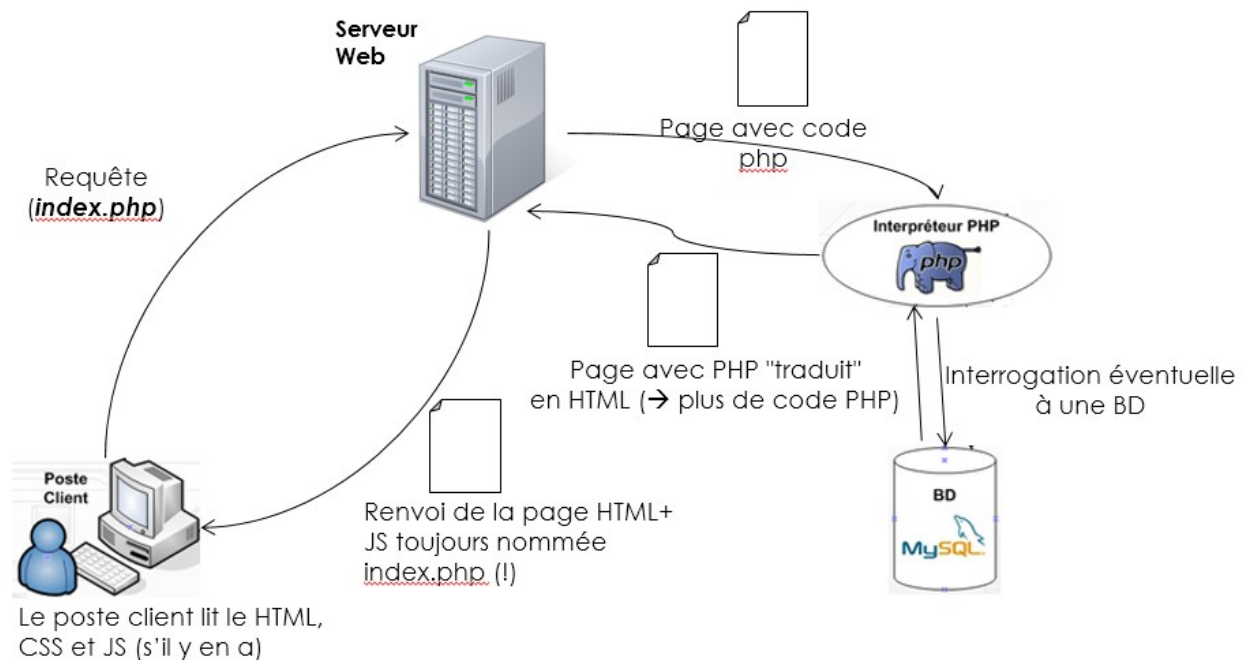
1.1.2 Petit historique

En 1994, Rasmus Lerdorf crée PHP qui à cette époque signifie Personal Home Page tools. Il souhaite relever les statistiques des visiteurs qui consultent son site web. Il écrit une bibliothèque en langage Perl, puis en langage C. Il publie ensuite son code source.

En 1997, Andi Gutmans et Zeev Suraski sont deux étudiants qui reprennent son travail et développent le moteur Zend Engine permettant d'utiliser PHP comme actuellement. L'arrivée de PHP5 en 2004 permet d'avoir la seconde génération du moteur Zend Engine 2. Il apporte notamment des nouvelles fonctionnalités tel que la programmation orienté objet, la gestion des erreurs et exceptions.

PHP7 arrive en fin 2015 avec un nouveau moteur Zend Engine, il est notamment plus performant et ajoute diverses fonctionnalités telles que la prise en charge du typage dans la déclaration des fonctions, des erreurs fatales converties en exception, etc.

1.2 Principe du PHP



1.3 Premier programme « Hello World ! »

Créer un fichier avec l'extension .php

```
<?php
// Faire un Hello World ! en PHP
echo "Hello World !";
print "Hello World !";
?>
```

Il est possible d'écrire directement du HTML dans un echo ou un print. Il est aussi possible d'inclure du PHP au HTML pour autant que le fichier soit .php

```
<?php
// Ecrire du HTML dans un echo
echo "<p>Hello World !</p>";
?>

<?php
/* Ecrire du PHP dans du HTML */
?>
<p>
```

(suite sur la page suivante)

(suite de la page précédente)

```
<?php
echo "Hello World !";
?>
</p>
```

Pour que la page soit validée W3C, il faut obligatoirement qu'elle soit complète.

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8" />
    <title>Mon premier Hello World ! en PHP</title>
  </head>
  <body>
    <?php
    // Ecrire un Hello World ! dans une page complète HTML
    echo "<p>Hello World !</p>";
    ?>
  </body>
</html>
```


Le principe de la programmation en PHP ne diffère pas de C#, il y a cependant parfois des particularités dans la syntaxe. C'est à la base un langage non typé. Cependant depuis la version 7, il est possible d'ajouter des types à certains éléments (méthodes, fonctions, variable de classe).

2.1 Variables et constantes

Les variables sont précédées d'un \$.

```
// Déclaration d'une variable
$captainsAge;
// Déclaration d'une variable avec une assignation
$number = 0;
```

Les constantes sont définies en majuscule, les mots séparés par des _ avec la fonction define ou const.

```
// Hors d'une classe
define("CAPTAINS_AGE", 54);

// A l'intérieur d'une classe
class foo {

    const CAPTAINS_AGE = 54;
}
```

2.2 Chaînes de caractères

Les chaînes de caractères se définissent également avec un \$. Il est possible d'utiliser les guillemets ou les simples guillemets.

```
$textWeather = "Aujourd'hui, il fera beau.";
$textWeatherNextWeek = 'La semaine prochaine, il neigera.';
```

La différence réside dans la substitution des variables. Si on intègre une variable dans une chaîne avec des guillemets, nous verrons la valeur de la variable. Si on fait de même dans une chaîne avec des simples guillemets, nous verrons le nom de la variable.

```
$alert = "Danger ! ";

$textWeather = "$alert Aujourd'hui, il fera beau.";
//Danger ! Aujourd'hui, il fera beau.
$textWeatherNextWeek = '$alert La semaine prochaine, il neigera.';
//$alert La semaine prochaine, il neigera.
```

Pour concaténer deux chaînes ensemble, l'opération est le point (.)

```
$textWeather = "Aujourd'hui, il fera beau.";
$textWeatherNextWeek = 'La semaine prochaine, il neigera.';

// En mettant les chaînes l'une après l'autre
echo $textWeather . $textWeatherNextWeek;
// Aujourd'hui, il fera beau.La semaine prochaine, il neigera.

// En ajoutant un espace entre les chaînes
echo $textWeather . " " . $textWeatherNextWeek;
// Aujourd'hui, il fera beau. La semaine prochaine, il neigera.
```

2.3 Opérateurs

➤ Mathématiques:

+	-	*	/	%
---	---	---	---	---

➤ De comparaison:

==	===	!=	!==	>	<	>=	<=
----	-----	----	-----	---	---	----	----

=== et !== comparent la valeur et le type

➤ Logiques :

and	&&	or		xor	!
-----	----	----	--	-----	---

➤ D'affectation :

=	++	--	+=	-=	*=	/=	%=
---	----	----	----	----	----	----	----

A retenir :

```
$textNumber = "1";  
$textNumber += 2; // La variable contiendra 3.
```

2.4 Conditions

```
if ($a > $b) {  
    echo "a est plus grand que b";  
} elseif ($a == $b) {  
    echo "a est égal à b";  
} else {  
    echo "a est plus petit que b";  
}
```

2.5 Switch

```
switch ($i) {  
    case 0:  
        echo "i égal 0";  
        break;  
    case 1:  
        echo "i égal 1";  
        break;  
    case 2:  
        echo "i égal 2";  
        break;  
    default:  
        echo "ce cas n'existe pas";  
}
```

2.6 While et Do While

```
// While  
$i = 1;  
while ($i <= 10) {  
    echo $i++;  
}  
  
// Do While  
$i = 0;  
do {  
    echo $i;  
} while ($i > 0);
```

2.7 For et Foreach

```
// For
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

// Foreach
$arr = array(1, 2, 3, 4);
foreach ($arr as $value) {
    $value = $value * 2;
}
```

2.8 Fonctions

```
function foo($arg_1, $arg_2, /* ..., */ $arg_n) {
    echo "Exemple de fonction.\n";
    return $retval;
}

// Depuis PHP7, il est possible de typer
function sum(int $a, int $b): int {
    return $a + $b;
}
```

CHAPITRE 3

Tableaux

En PHP, les tableaux sont des éléments importants, car ils permettent d'obtenir une structure ordonnée en clé => valeur. Ils sont considérés comme des dictionnaires, listes de données, collection, etc. C'est une différence par rapport au C# qui contient toutes ces structures.

3.1 Tableaux simples

Pour définir un tableau, le mot-clé array est utilisé. Les valeurs sont accessibles par leur indice.

```
// Définir un tableau vide
$newArray = array();

// Définir un tableau contenant des éléments
$colors = array("vert", "blanc", "rouge");

// Ajouter un élément dans un tableau existant
$colors[] = "jaune";

// Afficher un élément en particulier
echo $colors[2]; // rouge
```

3.2 Tableaux associatifs

Les tableaux associatifs ont la particularité d'être définis par une clé au lieu d'un indice. De plus, la syntaxe est clé => valeur.

```
// Définir un tableau vide
$newArray = array();

// Définir un tableau contenant des éléments
```

(suite sur la page suivante)

(suite de la page précédente)

```
$peopleAges = array("Paul" => 23, "Jacques" => 42);

// Ajouter un élément dans un tableau existant
$peopleAges["Jean"] = 35;

// Afficher un élément en particulier
echo $peopleAges["Jacques"]; // 42
```

3.3 Tableaux multidimensionnels

Nous avons vu les deux types de tableaux existants (indice et associatif). Il est possible de faire des tableaux multidimensionnels soit avec des indices, soit associatifs, soit le deux en fonction du besoin.

```
// Tableaux permettant de définir un carnet d'adresses
$addressBook = array(
    0 => array(
        "name" => "Eponge",
        "firstname" => "Bob",
        "tel" => 0987546378
    ),
    1 => array(
        "name" => "Neutron",
        "firstname" => "Jim",
        "tel" => 0384560924
    )
);

// Ajouter un élément par la suite
$addressBook[] = array(
    "name" => "Dalton",
    "firstname" => "Jeo",
    "tel" => 4728293745
);

// Afficher un élément
echo $addressBook[2]["name"]; // Dalton
```

3.4 Parcourir un tableau

Il est possible de parcourir un tableau avec tout type de boucles comme en C#, cependant la boucle foreach est très utile et efficace.

```
// Parcourir un tableau simple
$colors = array("vert", "blanc", "rouge");

foreach($colors as $color){
    echo $color;
}
```

```
// Parcourir un tableau associatif
$peopleAges = array("Paul" => 23, "Jacques" => 42);
```

(suite sur la page suivante)

(suite de la page précédente)

```
foreach($peopleAges as $name => $age) {  
    echo $name . " = " . $age . "<br>";  
}
```

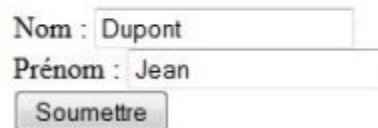
3.5 Quelques fonctions

Il existe plusieurs fonctions consacrées aux tableaux en PHP. [Liste du site de PHP.net](#)

Voici la liste des principales que nous allons utiliser : `var_dump()`, `print_r()`, `count()`, `explode()`, `end()`, `sort()`, `asort()`, `ksort()`, etc.

Pour permettre des interactions avec les utilisateurs d'un site Internet, le formulaire est essentiel. Il se compose d'un ou plusieurs champs à remplir et d'un bouton. Il est fait en HTML. Le JavaScript peut venir compléter le formulaire pour permettre sa validation ou de l'interaction (formulaire intelligent). Le PHP, quant à lui, permet de traiter les données ou de pré-remplir les champs (dans le cas d'un formulaire de modification).

Créer un nouveau compte



Nom : Dupont

Prénom : Jean

Soumettre

Code du formulaire ci-dessus

```
[début de la page]
<form method="post" action="checkConnection.php">
  <p>
    <label for="name">Nom : </label>
    <input type="text" name="lastname" id="name" />
  </p>
  <p>
    <label for="firstname">Prénom : </label>
    <input type="text" name="firstname" id="firstname" />
  </p>
  <p>
    <input type="submit" name="btnSubmit" value="Soumettre" />
  </p>
</form>
```

4.1 Balises de formulaire

form Définit le début et la fin du formulaire.

Attribut	Exemple	Définition
name	name= »connection »	Définit le nom du Formulaire
method	method= »post »	Définit la méthode d'envoi (POST ou GET)
action	action= »page.php »	Définit la page qui est exécutée au clic du bouton.

input Champ de saisie sur une ligne. Balise auto-fermante. Il faut lui préciser un type de données (text, password, radio, number, email, checkbox, hidden etc). Le type « submit » permet d'envoyer un formulaire dans le cas d'un bouton.

textarea Champ de saisie sur plusieurs lignes.

select Permet de définir une liste déroulante. Chaque élément de la liste est une « option ». Le nom se place sur la balise « select » et il est nécessaire d'ajouter un attribut « value » sur chaque option permettant de récupérer la sélection.

```
<select name="job">
  <option value="1">Paysagiste</option>
  <option value="2">Médecin</option>
  <option value="3">Bibliothécaire</option>
</select>
```

4.2 Envoi d'un formulaire

Pour qu'un formulaire s'envoie, il faut définir une méthode d'envoi dans la balise form, une action et avoir un bouton de type submit.

Par défaut, si la méthode n'est pas spécifiée, c'est la méthode GET qui est utilisée. Par défaut, si l'action n'est pas utilisée, c'est l'action # qui est utilisée (on reste sur la même page).

Cependant, il est recommandé de définir la méthode et l'action pour éviter des mauvaises surprises.

```
// Définir la méthode
<form method="post" action="page.php">

// Avoir un bouton de type submit
<input type="submit" value="Envoyer" />

ou

<button type="submit">Envoyer</button>
```

4.3 Traitement en GET

La méthode GET permet de passer les données du formulaire directement depuis l'URL de la page. C'est rarement utilisé dans la cadre d'un formulaire à remplir vu que les informations sont directement visible. C'est plutôt utilisé dans le cadre d'un formulaire de recherche ou pour atteindre une page avec des variables définies. Il ne faut pas oublier qu'il existe une limite de caractères dans l'URL, c'est pourquoi, cette méthode n'est pas des plus adaptée.

Pour récupérer le contenu des données passés en GET, une variable super-globale \$_GET existe. C'est un tableau associatif rempli avec les données.

Si dans un formulaire, il y a un champ pour le prénom nommé « `firstname` », il est possible de récupérer sa valeur ainsi :

```
<input type="text" name="firstname" value="toto" />

$_GET["firstname"]; // toto
```

L'URL ressemble à ceci :

```
http://nomdedomaine.ext/page.php?firstname=toto
```

4.4 Traitement en POST

Le méthode POST permet de passer les données du formulaire de manière masquée. Elles sont traitées dans l'en-tête du message HTTP, elles ne sont pas cryptées. C'est le même principe que pour le GET, c'est une super-globale, cette fois-ci appelée `$_POST` qui est sous forme de tableau associatif. `$_POST` n'a pas de limite, ce sont les réglages du serveur qui font foi.

```
<input type="text" name="firstname" value="toto" />

$_POST["firstname"]; // toto
```

En cas de formulaire qui contient des données sensibles, le protocole https est fortement recommandé.

4.5 Traitement des fichiers et images

Dans un formulaire, il est parfois utile de pouvoir ajouter des documents (pdf, word, etc.) ou des images (image de profil, capture d'écran, etc.). Ce n'est rien d'autre qu'un formulaire avec 2-3 options en plus.

Formulaire de téléchargement

```
<form action="uploadFile.php" method="post" enctype="multipart/form-data">
  <p>
    <label for="printscreen">Fichier à télécharger</label>
    <input type="file" name="printscreen" id="printscreen" />
  </p>
  <p>
    <input type="submit" value="Envoyer" />
  </p>
</form>
```

Pour télécharger un fichier ou une image, c'est obligatoirement la méthode POST qui est utilisée. En effet, c'est difficile de passer une telle quantité de donnée dans une URL.

De plus, un attribut supplémentaire à la balise form est ajouté (`enctype= »multipart/form-data »`) permettant de prendre en compte le traitement des images. Par défaut, les caractères sont encodés avant l'envoi (les espaces sont convertis en `+` et les caractères spéciaux transformé en ASCII) (`enctype= »application/x-www-form-urlencoded »`).

Pour récupérer les données du formulaire, nous connaissons la super-globale \$_POST. Cette dernière fonctionne très bien pour tous les champs simples. Mais pour récupérer les données d'un téléchargement, une autre super-globale est utilisée : \$_FILES.

Plusieurs informations provenant d'un fichier sont stockées, telles que son nom, son type, son code erreur, son nom temporaire et sa taille.

Dans le cas d'un formulaire en POST avec des champs simples et des champs de téléchargement, les champs simples seront traités avec un \$_POST[« nomDuChamp »] et les champs de téléchargement avec un \$_FILES[« nomDuChamp »].

J'ai inséré une image « capture.jpg » dans mon champ « printscreen », voici ce que je peux récupérer :

```
//capture.jpg (nom d'origine de mon fichier téléchargé)
$_FILES["printscreen"]["name"];

//image/jpeg (type de mon fichier)
$_FILES["printscreen"]["type"];

//xdhgqjd.jpg (nom que le serveur va attribuer à mon fichier pour éviter qu'il écrase
//un autre fichier avec le même nom)
$_FILES["printscreen"]["tmp_name"];

//0 (signifie que tout s'est bien passé)
$_FILES["printscreen"]["error"];

//37465 (taille du fichier en octets)
$_FILES["printscreen"]["size"];
```

Une fois que les données ont été récupérées via \$_FILES, il faut déplacer le fichier dans un répertoire du site web afin de le stocker. Une méthode de déplacement est utilisée qui a deux arguments : move_uploaded_file (fichier source, destination du fichier)

```
$source = $_FILES["printscreen"]["tmp_name"];

//La destination permet de définir non seulement le chemin du fichier, mais aussi
//un nouveau nom si nécessaire
$destination = "images/" . date("YmdHis") . $_FILES["printscreen"]["name"];

move_uploaded_file($source, $destination);
```

4.6 php.ini

C'est le fichier de configuration de PHP, chaque serveur en contient un. Il permet d'indiquer des directives, de fixer des valeurs ou d'activer des modules complémentaires. Voici les variables qui s'y rapportent :

file_uploads : Autorise ou non l'envoi de fichiers (ON / OFF)

upload_tmp_dir : Répertoire temporaire d'accueil du fichier

upload_max_filesize : Définit la taille maximale autorisée pour l'envoi d'un fichier (ex : 2Mo)

post_max_size : Définit la taille maximale pour l'envoi d'un formulaire en POST (tous les champs compris). (ex : 8Mo)

Tous ces paramètres sont accessibles depuis le fichier php.ini ou en utilisant la méthode phpinfo() dans un fichier php.

Les cookies sont des fichiers enregistrés sur l'ordinateur du client (le visiteur). Ils permettent de stocker des informations et de les lire. C'est un système qui permet d'identifier et de suivre les visiteurs. Par exemple, vous pouvez stocker dans un cookie le nom du visiteur de votre site demandé via un formulaire et ainsi, la prochaine fois qu'il revient, vous pouvez lui souhaiter le bonjour avec son nom.

5.1 Ecrire un cookie

Une fonction `setcookie()` permet de créer un cookie. Il comprend en général les 3 premiers paramètres (son nom, sa valeur et sa date d'expiration). Cependant, d'autres paramètres sont disponibles.

Pour chaque valeur à stocker, un cookie est créé. Donc, si on désire stocker le nom et le navigateur utilisé de notre visiteur, deux cookies seront faits.

La fonction `setcookie` doit être placée avant d'appeler tout code html dans la page.

```
// Stocker le nom (qui vient d'un formulaire)
$name = "Alfred";
setcookie("name", $name, time() + 365*24*3600);

// Stocker le navigateur utilisé
$browser = $_SERVER["HTTP_USER_AGENT"];
setcookie("browser", $browser, time() + 365*24*3600);

// Si vous souhaitez que votre cookie ne soit pas accessible via JavaScript,
// vous devez activer l'option httpOnly
setcookie("name", $name, time() + 365*24*3600, null, null, false, true);
```

5.2 Afficher un cookie

Une super-globale existe pour utiliser le cookie : `$_COOKIE[« nom du cookie »]`;

```
$_COOKIE["name"]; // Alfred
```


Jusqu'à présent, nous avons utilisé des variables ou constantes accessibles uniquement sur la page courante. Il est parfois nécessaire de pouvoir stocker certaines informations pour les reprendre plus tard, sur une autre page du site par exemple. Les sessions sont le moyen de conserver des valeurs à travers les pages d'un site sans pour autant avoir un formulaire en GET ou POST en entrée.

6.1 Fonctionnement

Pour que les sessions fonctionnent sur les pages d'un site, il faut obligatoirement démarrer les sessions (et ce, sur chaque page où les sessions seront utilisées.) Cette méthode doit être déclarée au début de chaque page (avant l'en-tête html).

```
session_start(); // Démarrer le système de session
```

Dès à présent, quand un visiteur arrivera sur votre site, une variable de session unique lui sera attribuée. Cette dernière est un numéro en hexadécimal très grand appelé « ID de session ». PHP se charge de transmettre ce numéro de page en page en le stockant dans un cookie.

6.2 Création

Maintenant que la session est démarrée, il est possible de créer autant de variables de session que nécessaire. Pour rappel, ce sont des variables qui vont pouvoir être utilisées sur n'importe quelles pages du site (pour autant que session_start() y figure). Pour créer une variable de session, une super-globale de type tableaux associatifs est utilisée.

```
$_SESSION["isConnected"] = 1; // Créer la variable de session isConnected et affecte 1
$_SESSION["firstname"] = "Kara"; // Créer firstname et lui affecte un prénom
```

6.3 Accès aux variables

```
// Contrôler si une variable existe
if(isset($_SESSION["isConnected"]))

// Voir toutes les variables de session qui existent
var_dump($_SESSION);

// Afficher le contenu d'une variable de session
echo $_SESSION["firstname"];
```

6.4 Suppression

Les variables de session se détruisent quand le visiteur part du site Internet. Mais il est difficile de savoir clairement quand le visiteur est parti. Va-t-il fermer le navigateur, seulement changer d'onglets ? Pour détruire les variables de session existantes, il existe deux solutions. Soit une fonction de déconnexion est créée, soit le timeout du serveur pour les sessions est utilisé.

```
// Se déconnecter - supprime l'ensemble des variables de session
session_destroy();

// Supprimer une variable de session en particulier
unset($_SESSION["firstname"]);

// Dans le fichier php.ini
session.cookie_lifetime = 0 // 0 indique jusqu'à ce que le navigateur est redémarré
```

CHAPITRE 7

Gérer les mots de passe

Dans le cas d'un formulaire d'authentification, le mot de passe doit être géré correctement. Un mot de passe utilisateur ne doit jamais être stocké en clair que ce soit dans un fichier ou dans une base de données. Il sera haché avant le stockage et une fonction de contrôle du mot de passe sera utilisée.

7.1 Cryptage VS Hachage

Pour rappel, le cryptage permet un chiffrement basé sur un algorithme et il est possible de retrouver l'origine (donc décryptage). Le hachage, quant à lui, se base aussi sur un algorithme et compose une empreinte de longueur fixe, irréversible. Il est impossible de retrouver l'origine.

De plus, dans le hachage, il est possible d'ajouter un sel pour sécuriser d'avantage l'empreinte et rendre de plus en plus difficile la création de rainbow table.

Il est recommandé que le mot de passe d'origine suive une policy (ex : chiffre, minuscule, majuscule, caractères spéciaux, etc.).

7.2 Hacher un mot de passe

PHP a une méthode permettant de hacher un mot de passe en utilisant un algorithme avec sel intégré.

```
password_hash("Mot de passe à hacher", algorithme à utiliser);  
password_hash("toto!1234", PASSWORD_BCRYPT);  
// $2y$10$UqpPCefjmclTg7X7VCr/Ke/n3o58d9VJM50DWqtrIXp8XJD3R4/8G
```

7.3 Vérification d'un mot de passe

Lors d'un formulaire de connexion, l'utilisateur rentrera toujours son mot de passe en clair. De notre côté, nous aurons stocké seulement l'empreinte. Pour les comparer, une fonction de PHP doit être utilisée.

```
$passwordUser = "toto!1234";  
$hash = "$2y$10$UqpPCefjmc1Tg7X7VCr/Ke/n3o58d9VJM50DWqtrIXp8XJD3R4/8G";  
  
password_verify($passwordUser, $hash); // Retourne un booléen
```

POO (Programmation Orienté Objet)

Dans cette partie, vous trouverez principalement la syntaxe de la programmation orienté objet.

8.1 Classe

Déclaration d'une classe avec deux propriétés, un constructeur, des getter/setter et une méthode.

```
class Car {  
  
    // Variables de classe  
    protected $color; // couleur  
    private $vinNumber; // numéro de chassis  
  
    /**  
     * Constructeur  
     * @param $vinNumber  
     */  
    function __construct($vinNumber) {  
        $this->setVinNumber($vinNumber);  
    }  
  
    /**  
     * Getter color  
     * @return string  
     */  
    public function getColor() {  
        return $this->color;  
    }  
  
    /**  
     * Setter color  
     * @param $color  
     */  
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

public function setColor($color){
    $this->color = $color;
}

/**
 * Getter vinNumber
 * @return string
 */
public function getVinNumber(){
    return $this->vinNumber;
}

/**
 * Setter vinNumber
 * @param $vinNumber
 */
private function setVinNumber($vinNumber){
    $this->vinNumber = $vinNumber;
}

/**
 * Méthode permettant de démarrer la voiture
 */
public function start(){
    // Instruction
}
}

```

8.2 Héritage

Le mot-clé `extends` doit être utilisé. Un seul héritage à la fois est autorisé.

```

// Création d'une classe parente
class Vehicle {

    private $numberOfSeats;

    /**
     * Getter numberOfSeats
     * @return string
     */
    public function getNumberOfSeats(){
        return $this->numberOfSeats;
    }

    /**
     * Setter numberOfSeats
     * @param $numberOfSeats
     */
    private function setNumberOfSeats($numberOfSeats){
        $this->numberOfSeats = $numberOfSeats;
    }

}

```

(suite sur la page suivante)

(suite de la page précédente)

```
// Héritage de la classe véhicule à la classe voiture
class Car extends Vehicle {}
```

8.3 Interface

Le mot-clé implements doit être utilisé. Il est possible d'implémenter plusieurs interfaces.

```
// Création d'une interface de mouvement
interface iMovable {
    public function turnRight();
    public function turnLeft();
}

// Implémentation de l'interface
class Car implements iMovable {}
```


MVC (Modèle - Vue - Contrôleur)

9.1 Concept d'architecture

L'architecture MVC est découpé en 3 zones :

- M pour Modèle qui contient les données à afficher.
- V pour Vue qui contient la présentation des éléments à afficher (interface graphique notamment).
- C pour Contrôleur qui contient la logique des actions effectuées par l'utilisateur. Chaque Vue est associée à un Contrôleur.

Principe du MVC

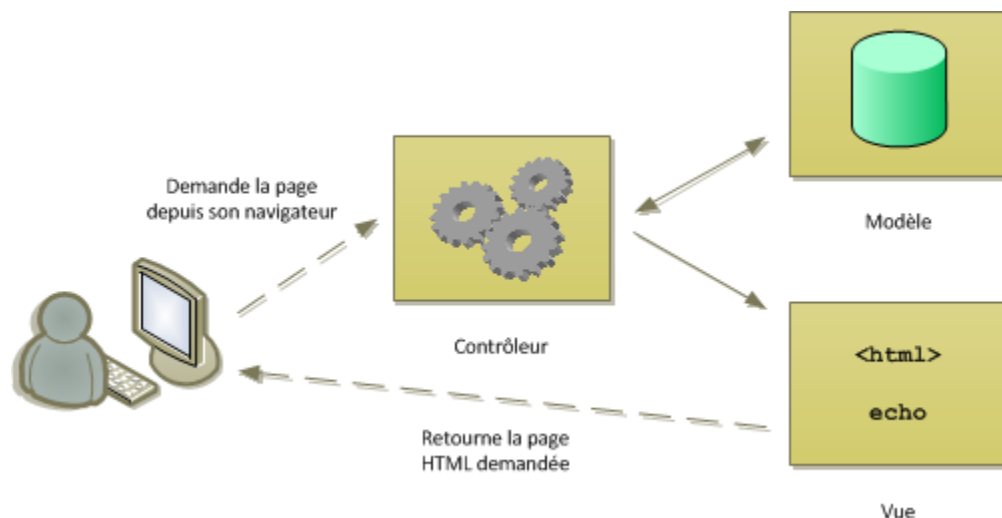


Fig. 1 – Source : Openclassroom MVC

Le client demande une page depuis son navigateur. L'architecture du site est faite en MVC (Modèle-vue-contrôleur). De ce fait, le client en cliquant sur une page va appeler sans le savoir le contrôleur de la page. Ce dernier va interagir avec le modèle pour récupérer les données nécessaires à la page demandée. Le travail du modèle est d'obtenir les

informations, souvent dans une base de données (par exemple : MySQL). Une fois, les données obtenues, il les renvoie au contrôleur. Les données sont maintenant en possession du contrôleur. Il va pouvoir les traiter au besoin et surtout les envoyer à la vue. Cette dernière se charge de la mise en forme des données sur la page.

9.2 Explication d'un modèle MVC basique

Téléchargement de l'application

Le PHP tourne sur des serveurs de type Linux. Il est possible de coder sur toutes les plateformes actuelles (Windows, Linux et Mac) en utilisant un serveur Web local. Pour Windows, le serveur utilisé sera uWamp : [uWamp](#).

Quelle que soit la page que l'on souhaite accéder, tout est dirigé par la page « index.php » se trouvant à la racine. Elle instancie un nouvel objet qui va permettre à son tour d'instancier le bon contrôleur et l'action pour l'affichage de la page.

Le contrôleur a pour but de chercher les données et de les transmettre à la vue, en utilisant la bonne action. Le modèle va chercher les données dans les fichiers fournis et la vue va afficher les données.

Dans les méthodes « display » et « listAction » de « CustomerController », certains commentaires ont été ajoutés afin d'expliquer le code.

Arborescence

Le fichier « index.php »

Dans cette architecture, tout démarre depuis la page « index.php ». L'URL sera toujours composée d'un contrôleur et d'une action. Par exemple, la page « liste des clients » est demandée. L'URL sera : <http://adresseDuSite/index.php?controller=customer&action=list>

Déroulement :

Au moment où l'utilisateur saisit cette URL, la page « index.php » est appelée. Un objet de type « MainController » est instancié et la méthode « dispatch » est appelée. Cette dernière permet de récupérer le contrôleur (dans l'URL) et l'action (dans l'URL). Elle appelle la méthode « menuSelected » qui va instancier le contrôleur en rapport avec celui demandé dans l'URL, puis pour finir va construire la vue qui sera affichée. Dans la méthode « viewBuild » le contrôleur instancié va appeler la bonne action grâce à la méthode « display ».

Le dossier « controller »

Il contient un contrôleur principal et tous les contrôleurs pour appeler les actions spécifiques. Ces derniers héritent du contrôleur principal. Chaque contrôleur sert à atteindre la bonne action permettant de construire la page que l'utilisateur aura appelé.

Le dossier « model »

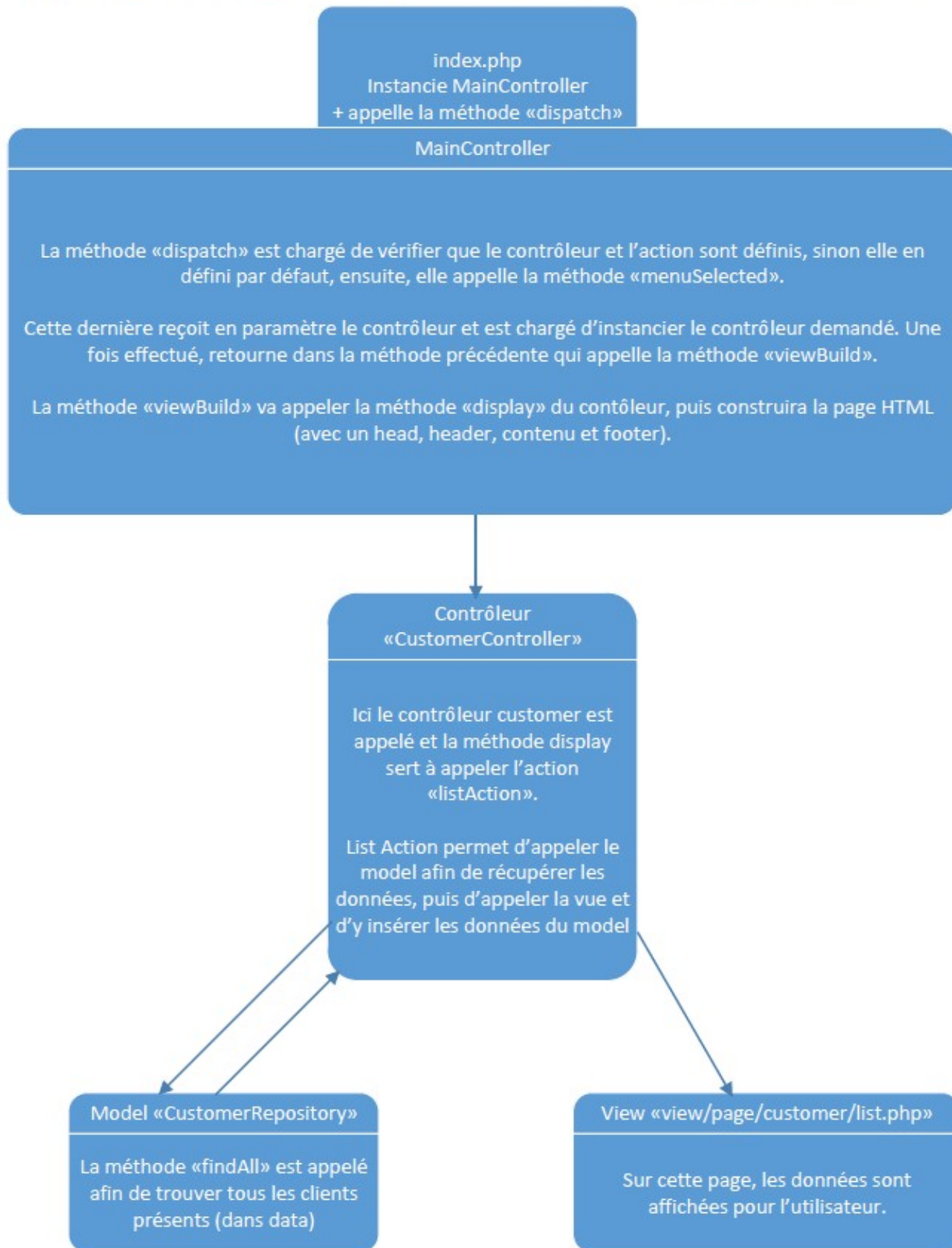
Il contient un fichier « Entity.php », ce dernier est l'interface qui désigne la ou les méthodes qui seront obligatoirement définies. Il a également des fichiers « Repository » par contrôleur qui implémente l'interface. Chaque repository contiendra autant de méthodes que nécessaire pour accéder aux données. Par exemple, la méthode « findAll() » permet de récupérer toutes les données et la méthode « findOne(\$id) » permet de repérer les informations pour un élément spécifique.

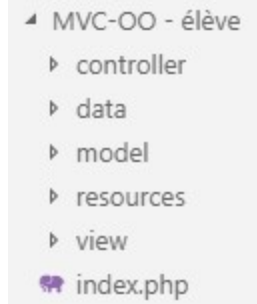
Le dossier « view »

A la racine de ce dossier se trouvent les fichiers généraux pour construire la page, à savoir head.php, header.php, footer.php et menu.php. Dans le dossier « page », il y aura un dossier par contrôleur et ensuite, une page par action. Exemple : si la page pour lister tous les clients est demandée, ce sera la view : /view/page/customer/list.php qui sera appelée.

Le dossier « data »

La page «Liste des clients» est demandée par l'utilisateur : `index.php?controller=customer&action=list`





Ce dossier sert à gérer les données de l'application. A terme, il sera remplacé par une autre gestion des données, telles qu'une base de données MySQL ou équivalent. Il y a deux fichiers. Le premier qui contient la liste des clients et le second, la liste des factures.

Le dossier « resources »

C'est le dossier qui permet de stocker tout ce qui concerne le CSS, les polices particulières, les images du design du site et le JavaScript.

9.3 Explication d'un framework MVC existant

```
<?php
// Faire un Hello World ! en PHP
echo "Hello World !";
print "Hello World !";
?>
```

CHAPITRE 10

PHPUnit

PHPUnit est une librairie permettant de réaliser des tests de votre code PHP. (même principe que vous avez vu en C#)

Pour installation, aller directement sur le site de [PHPUnit](#)

Exemple

Lors de l'exercice des commandes, dans la classe Order, nous allons tester la méthode public computeVAT().

```
/**
 * calcule de la TVA
 *
 * @return float
 */
public function computeVAT() {
    $sumTVA = $this->sum() * 0.077;
    return $this->special_round($sumTVA, 0.05);
}
```

```
use PHPUnit\Framework\TestCase;

require __DIR__ . '/../src/Product.php';
require __DIR__ . '/../src/Customer.php';
require __DIR__ . '/../src/Order.php';

class OrderTest extends TestCase {

    /**
     * test sur le calcul de la TVA
     */
    public function testComputeTVAWithOneProduct() {
        $product = new Product('Un produit', 'Une description', 40.0);
        $customer = new Customer();

        $order = new Order($customer, $product);
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

        $this->assertSame(3.1, $order->computeVAT());
    }
}

```

L'utilisation de PHPUnit se fait via une ligne de commande. Il suffit de lancer les tests.

```

# Pour lancer un test sur une classe spécifique
./vendor/bin/phpunit --bootstrap vendor/autoload.php tests/OrderTest

# Rendu dans le cas où le test ne passe pas
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

F                                     1 / 1 (100%)

Time: 250 ms, Memory: 4.00 MB

There was 1 failure:

1) OrderTest::testComputeTVAVWithOneProduct
Failed asserting that 3.1 is identical to 1.1.

C:\UwAmp\www\133CommandeAvecTest\tests\OrderTest.php:22

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.

# Rendu dans le cas où le test passe
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

.                                     1 / 1 (100%)

Time: 172 ms, Memory: 4.00 MB

OK (1 test, 1 assertion)

```

```

# Pour lancer tous les tests du dossier
./vendor/bin/phpunit --testdox tests/

# Rendu dans le cas où le test ne passe pas
PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

Order
  Compute t v a with one product
    Failed asserting that 3.1 is identical to 1.55.

    C:\UwAmp\www\133CommandeAvecTest\tests\OrderTest.php:18

Time: 242 ms, Memory: 4.00 MB

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.

# Rendu dans le cas où le test passe

```

(suite sur la page suivante)

(suite de la page précédente)

PHPUnit 8.0.0 by Sebastian Bergmann and contributors.

Order

✓ Compute t v a with one product

Time: 172 ms, Memory: 4.00 MB

OK (1 test, 1 assertion)

CHAPITRE 11

Commentaires

Tout comme en C#, il existe les commentaires classiques en PHP et les commentaires pouvant servir à la génération automatique de documentation.

11.1 Classique

```
// Permet d'écrire des commentaires sur une ligne

# Permet d'écrire des commentaire sur une ligne, rarement utilisé

/*
Permet d'écrire des
commentaire sur
plusieurs lignes
*/
```

11.2 PHPDoc

```
/**
 * Somme de deux nombres
 *
 * @param a premier nombre
 * @param b deuxième nombre
 *
 * @return int
 */
function sum(int a, int b) : int{
    return a + b;
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
/** @var string $name */  
$name; // Permet de spécifier le type d'une variable, pratique pour les éditeurs  
  
/** @var Car $opel */  
$opel = new Car();
```

12.1 1. Exercice introduction

12.1.1 1.1 Rappel

Créer une page HTML. La mise en page en CSS est facultative. La page HTML doit contenir :

- Un titre de niveau 1
- Un titre de niveau 2
- Une liste à puce
- Une image
- Un mini formulaire avec nom, prénom et bouton d'envoi (qui ne fonctionne pas)
- Un trait
- Un paragraphe
- Une adresse mail

12.1.2 1.2 Hello World !

Créer votre premier fichier PHP pour afficher « Hello World ! ». La page doit être validée d'un point de vue HTML.

12.2 2. Syntaxe

12.2.1 2.1 Condition

Créer un nombre aléatoire en utilisant la fonction `rand()`. Le nombre doit être compris entre 1 et 25. Si le nombre est inférieur à 18, afficher « Vous êtes mineur », sinon afficher « Vous êtes majeur ».

12.2.2 2.2 Condition

Créer un nombre aléatoire compris entre 1 et 80. Selon le nombre, afficher une des phrases suivantes :

- Si le nombre est entre 1 et 17, « Vous êtes mineur ».
- Si le nombre est entre 18 et 25, « Vous êtes un jeune adulte ».
- Si le nombre est entre 26 et 65, « Vous êtes un adulte ».
- Si le nombre est entre 66 et 80, « Vous êtes senior ».
- Sinon, votre âge n'existe pas.

12.2.3 2.3 Boucle

Créer un code pour trouver les 10 premiers nombres de la suite de Fibonacci.

Pour rappel, la suite de Fibonacci est 0,1,1,2,3,5,8,...

Les deux premiers nombres sont 0 et 1. Ensuite, chaque nombre est la somme des deux précédents.

12.2.4 2.4 Fonction

Créer une fonction pour calculer un nombre à la puissance désirée.

Par exemple : le nombre 2 à la puissance 3 donne le résultat 8.

La fonction aura deux paramètres d'entrée. Un paramètre pour le nombre et un autre pour la puissance. La fonction retournera le résultat.

12.3 3. Tableaux

12.3.1 3.1 Création de tableaux

Créer un tableau simple contenant les jours de la semaine. Les jours de la semaine sont lundi, mardi, mercredi, jeudi, vendredi, samedi et dimanche.

Créer un second tableau associatif contenant la liste des mois avec leur saison. Par exemple : janvier est en hiver, août est en été.

Afficher les tableaux en utilisant les fonctions `var_dump()` et `print_r()`.

12.3.2 3.2 Parcourir les tableaux

Parcourir le tableau des jours de la semaine avec une boucle `for`.

Parcourir le tableau des mois et saisons avec une boucle `foreach`.

12.3.3 3.3 Tableau multidimensionnel

Créer un tableau multidimensionnel pour stocker les notes de module pour un élève.

Par exemple : Basile => 100 => 5.5, 301 => 4.0, 226 => 6.0

Lucien => 100 => 6.0, 301 => 5.0, 226 => 2.5

Ensuite parcourir le tableau avec des boucles `foreach`. Et afficher chaque élève avec ses résultats.

12.4 4. Formulaire

12.4.1 4.1 Création d'un formulaire

Créer un formulaire pour rapporter les bug avec les champs suivants :

- Un champ pseudo
- Un champ de texte pour décrire le problème
- **Une liste déroulante pour définir l'importance du problème avec les éléments suivants :**
 - Choisir
 - Basse
 - Moyenne
 - Prioritaire
- Deux boutons radios pour définir si le ticket est privé ou public.
- Une case à cocher pour recueillir le consentement de l'utilisateur pour le traitement de ses données.
- Un bouton d'envoi

12.4.2 4.2 Traitement du formulaire

Lors de l'envoi du formulaire précédent, reprendre chaque champ et les afficher sur une page de résumé. La vérification des champs se fera dans l'exercice suivant.

12.4.3 4.3 Vérification de chaque champ du formulaire

Avant d'afficher les résultats, vérifier chaque champ selon cette logique :

- Pseudo : Seulement du texte et des chiffres, non vide
- Champ de texte : non vide
- Importance : 1 de sélectionné, mais ne doit pas être « choisir »
- Bouton radio : 1 de sélectionné
- Case à cocher : doit être cochée

Si un de ces champs est incorrect, un message d'erreur apparaît à la place de l'affichage.

12.4.4 4.4 Transfert de fichier

Créer un formulaire avec un champ pour ajouter un fichier et un bouton d'envoi. Lors de l'envoi, récupérer le fichier de l'utilisateur et stocker le fichier dans le dossier image de votre site. Attention, seul les jpg et les png sont acceptés. Avant de le sauvegarder dans votre répertoire, donner un nom unique au fichier.

12.5 5. Cookie

Créer un formulaire demandant le login de l'utilisateur et un bouton « Soumettre ». Une fois le bouton cliqué, créer les cookies suivants :

- Un cookie permettant de stocker le pseudo
- Un cookie permettant de stocker le navigateur utilisé
- Un cookie permettant de stocker le nombre de fois que le formulaire a été soumis pour le poste client

Les cookies sont uniquement créés/modifiés si le formulaire a été soumis.

12.6 6. Session

12.6.1 6.1 Authentification

Authentification

Pseudo Mot de passe

[Entrer ...](#)

Créer un formulaire de connexion comme ci-dessus et un lien « Entrer... » en dessous du formulaire menant sur « profile.php ». Le contenu de la page « profile.php » sera expliquée plus bas.

Créer un tableau en PHP avec ces informations :

- toto@toto.ch hello
- admin@toto.ch super

L'adresse mail sert de login et l'autre texte de mot de passe.

En appuyant sur le bouton « ok », il faut créer une connexion. Le but est de comparer les informations de l'utilisateur. Donc, le login et le mot de passe doivent être identiques aux informations contenues dans le tableau.

Utiliser la session pour garder la connexion active. Pour contrôler si la connexion est active, appuyer sur le lien « Entrer... », le texte « La connexion est active » doit être affiché. Si la connexion n'a pas été faite, le texte « Aucune permission de visualiser cette page sans être connecté » doit être affiché.

Une fois fonctionnelle, modifier le tableau et encoder les mots de passe via la méthode PHP password_hash() et contrôler que votre connexion fonctionne toujours.

12.7 7. POO

Créer un petit programme capable de passer des commandes.

Schéma UML

Rendu

12.8 8. MVC

Afficher les factures

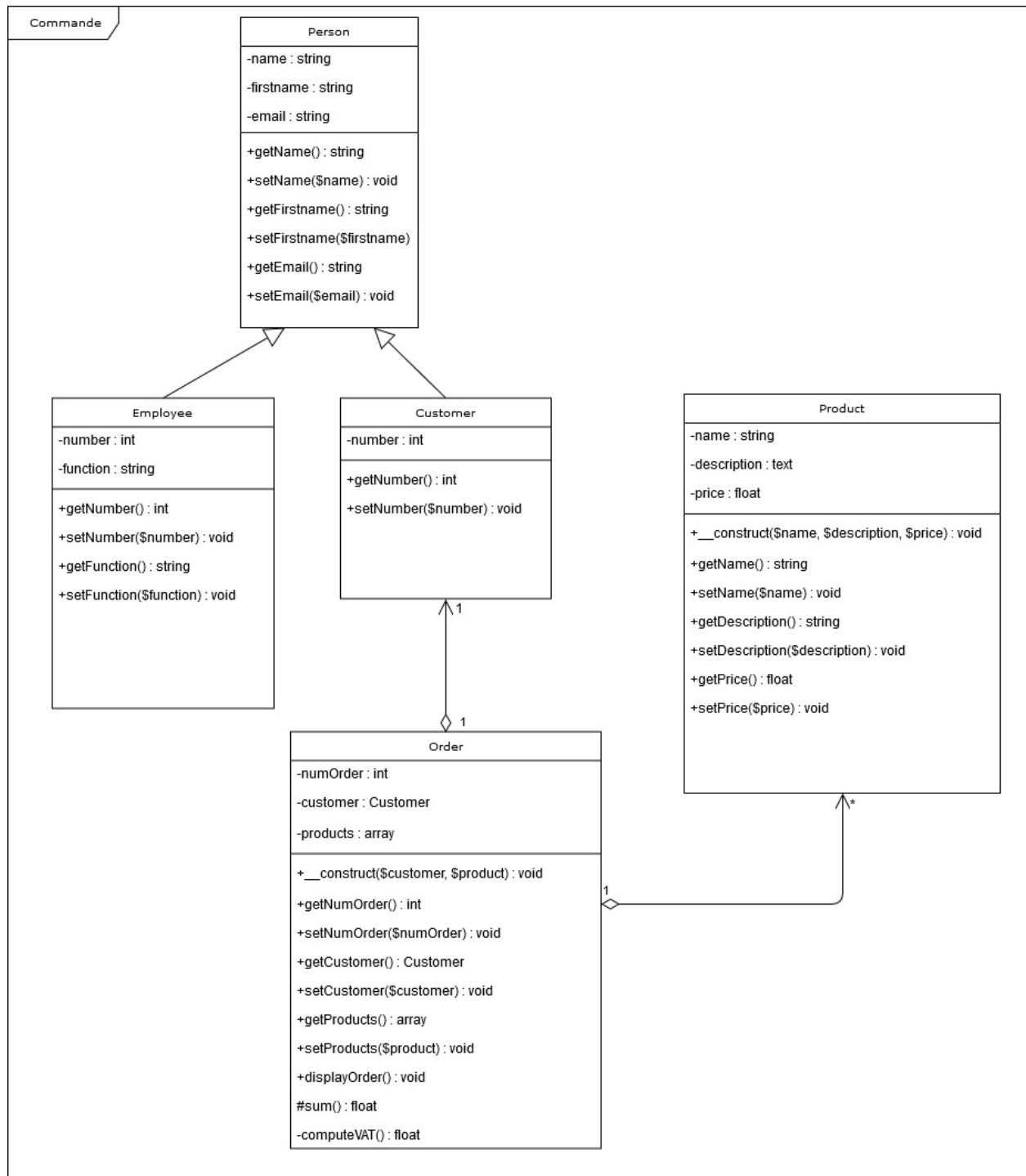
Remarques préalables :

- Ne pas hésiter à regarder dans les fichiers déjà créés pour s'en inspirer.
- Ne pas oublier d'inclure les fichiers nécessaires.

Le but de l'exercice est d'afficher les informations de la page « Liste des factures ». Le rendu devra ressembler à cette copie d'écran :

Dans le menu, le lien avec l'URL existe déjà : index.php?controller=invoice&action=list

1. Créer le contrôleur « InvoiceController » dans le dossier « controller ». Ajouter les actions (sous forme de méthode) « display » et « list », ainsi que le code adéquat. Dans l'action « list », récupérer les données des factures en utilisant le repository (voir point 2).



Client : Dujardin Paul
Numéro de commande : 337991
Liste des produits :
Windows 399.5
Excel 399.9
Total de la commande : 799.4
Total TVA : 61.55
Montant final à payer : 860.95

CRM - Clients et factures

Accueil	Liste des clients	Liste des factures	Contact
---------	-------------------	--------------------	---------

Liste des factures

Numéro	Quantité	Prix	Contact (ID)
45674	3	568.65	3
45675	4	359.20	4
45676	1	63.70	1
45677	4	1872.20	3
45678	2	392.15	6
45679	6	452.40	8
45680	2	387.20	3
45681	1	39.90	2

ETML 2019

2. Créer le repository « InvoiceRepository » dans le dossier « model ». Ajouter la méthode nécessaire pour récupérer toutes les données concernant les factures (du fichier data/invoices.php).
3. Créer le dossier « invoice » dans le dossier « view/page/ », puis le fichier « list.php » à l'intérieur. Ajouter le html nécessaire pour faire un affichage de chaque facture selon la copie d'écran.
4. Enfin, Modifier le fichier « index.php » de la racine pour ajouter votre nouvelle modification. Dans la méthode « menuSelected » de la classe « MainController » de ce fichier, ajouter un cas dans le switch permettant d'instancier un objet « InvoiceController ».

Pour les plus avancés

Compléter votre vue pour les factures en y insérant le nom et prénom du client.

CRM - Clients et factures

Accueil	Liste des clients	Liste des factures	Contact
---------	-------------------	--------------------	---------

Liste des factures

Numéro	Quantité	Prix	Contact (ID)	Contact (Nom)
45674	3	568.65	3	Zappelo Antoine
45675	4	359.20	4	Roland Françoise
45676	1	63.70	1	Dupond Jean
45677	4	1872.20	3	Zappelo Antoine
45678	2	392.15	6	Favre Loïc
45679	6	452.40	8	Beauford Valérie
45680	2	387.20	3	Zappelo Antoine
45681	1	39.90	2	Monney Sophie

ETML 2019

12.9 9. PHPUnit

CHAPITRE 13

Sources

13.1 Livre

— Exemple, Auteur, Edition ENI, ISBN 9782746092143

13.2 Site

— Exemple