

IUT R&T - M2105 développement web

----- Cours TP TD -----

Bruno Guégan
BG



IUT R&T - M2105 développement web: ----- Cours TP TD -----

par Bruno Guégan et

Copyright © **2004-2019**

Résumé

Le module M2105 permet d'appréhender la réalisation d'application web en HTML/CSS3/PHP/MySQL au travers d'une pédagogie de projet.

Pour toute interrogation ou suggestion au sujet de ce document : `<bruno.guegan@rt-iut.re>`

La reproduction exacte et la distribution intégrale de ce document sont permises sur n'importe quel support d'archivage, pourvu que cette notice soit préservée.

Table des matières

1. Les bases de PHP	1
1.1. Évaluation	2
1.1.1. TP1 : bases en PHP (exercices progressifs)	3
1.1.1.1. Objectifs et déroulement	3
1.1.1.2. Exercice 0 : configuration	3
1.1.1.3. Exercice 1	3
1.1.1.4. Exercice 2	4
1.1.1.5. Exercice 3	6
1.1.1.6. Exercice 4	8
1.1.1.7. Exercice 5	9
1.1.1.8. Exercice 6	9
1.1.2. TP2 : bases en PHP	13
1.1.2.1. Objectifs et déroulement	13
1.1.2.2. Exercice 1 : configuration	13
1.1.2.3. Exercice 2	13
1.1.2.4. Exercice 3	14
1.1.2.5. Exercice 4	14
1.1.2.6. Exercice 5	15
1.1.2.7. Exercice 6	15
1.1.3. TP3 : bases en PHP (tableaux & formulaires)	18
1.1.3.1. Objectifs et déroulement	18
1.1.3.2. Exercice 1 : tableau simple	18
1.1.3.3. Exercice 2 : différents type d'affichage	18
1.1.3.4. Exercice 3 : tableau une simple boucle	19
1.1.3.5. Exercice 4 : tableau entré par l'utilisateur	19
1.1.3.6. Exercice 5 : l'utilisateur sélectionne une entrée d'un tableau	19
1.1.3.7. Exercice 6 : manipulation de tableau	20
1.1.3.8. Exercice 7 : tableau multidimensionnel	21
1.1.3.9. Exercice 8 : tri à bulles	21
2. Sessions et cookies	23
2.1. Cours	25
2.1.1. À propos	26
2.1.1.1. Objectifs	26
2.1.1.2. Licence	26
2.1.2. Cookies	27
2.1.2.1. Généralités	27
2.1.2.1.1. Définition	27
2.1.2.1.2. Stockage	27
2.1.2.1.3. Caractéristiques	27
2.1.2.1.4. Les différents champs	27
2.1.2.1.5. Exemple fichier cookies	28
2.1.2.2. Écriture d'un cookie	29
2.1.2.3. Lecture d'un cookie	30
2.1.2.4. Effacement d'un cookie	31
2.1.3. Sessions	32
2.1.3.1. Définition	32
2.1.3.2. Initialisation	32
2.1.3.3. Les fonctions de gestion	32
2.1.3.3.1. session_start()	33
2.1.3.3.2. session_register(nom1, nom2, ...)	33
2.1.3.3.3. session_destroy()	33
2.1.3.3.4. session_unset()	33
2.1.3.4. Les fonctions d'accès aux variables	34
2.1.3.4.1. session_is_registered()	34
2.1.3.4.2. session_unregister()	34
2.1.3.4.3. session_name([opt nom])	34

2.1.3.4.4. session_id([opt id])	35
2.1.3.5. Exemples d'implémentation	35
2.1.3.5.1. Création d'un espace membre	35
2.1.3.5.2. Compteur de visiteurs online	36
2.1.3.6. Sérialisation des données	36
2.2. Évaluation	38
2.2.1. TP1 Série PHP	39
2.2.1.1. Exercice 1	39
2.2.1.2. Exercice 2	39
2.2.1.3. Exercice 3	40
2.2.1.4. Exercice 4	40
2.2.1.5. Exercice 5	40
2.2.1.6. Exercice 6	41
2.2.1.7. Exercice 7	41
2.2.2. TP2 les Cookies	42
2.2.2.1. Exercice 1	42
2.2.2.2. Exercice 2	42
2.2.2.3. Exercice 3	42
2.2.2.4. Exercice 4	42
2.2.3. TP3 les Sessions	44
2.2.3.1. Exercice 1	44
2.2.3.2. Exercice 2	44
2.2.3.3. Exercice 3	44
2.2.3.4. Exercice 4	44
2.2.4. TP4 synthèse	46
2.2.4.1. Généralités	46
2.2.4.2. Exercice 1 : les statistiques d'un site	46
2.2.4.2.1. Sujet	46
2.2.4.2.2. Etude du problème	47
2.2.4.2.3. Méthode de test	47
2.2.4.2.4. Affichage des statistiques	47
2.2.4.3. Exercice 2 : un logiciel de chat	47
2.2.4.3.1. Sujet	47
2.2.4.3.2. Etude du problème	47
2.2.4.3.3. Améliorations	47
2.2.5. TP5 : Gestion d'un caddie en PHP	48
2.2.5.1. Objectifs	48
2.2.5.2. L'art des choix	48
2.2.5.3. Utilisation des sessions	48
2.2.5.3.1. Définition des actions	49
3. Évaluation TD & TP du module M2105	53
3.1. Évaluation	54
3.1.1. TD1 : Analyse d'un système de gestion de commandes	55
3.1.1.1. Etude du système de gestion de commandes (MCD)	55
3.1.1.2. Réalisation de l'application PHP	56
3.1.2. TD2 : Incorporer un framework CSS & javascript au système de gestion de commandes	59
3.1.2.1. Installation de Foundation	59
3.1.2.2. Configuration de Foundation	60
3.1.3. TD3 : Amélioration et finalisation du système de gestion de commandes	64
3.1.3.1. Amélioration du code PHP	64
3.1.3.2. Utilisation d'Ajax	65
3.1.3.3. Ce qui reste à faire !	70
3.1.4. TP1 : <i>Le Grand Hôtel</i> - MCD et requêtes SQL	72
3.1.4.1. Modifier le modèle MCD (AnalyseSI)	73
3.1.4.2. Définir des requêtes SQL	74
3.1.5. TP2 : Foundation intégration	76
3.1.5.1. Quelques points techniques	78
3.1.6. TP3 : Gestion des comptes utilisateur	81

3.1.6.1. Quelques points techniques	82
3.1.7. TP4 : La réservation	85
3.1.7.1. L'algorithme permettant de renseigner la table 'Lignefacture'	86
3.1.7.1.1. Votre travail	88
3.1.8. TP5 : Choix de la chambre et facturation	90
3.1.8.1. Choix de la chambre	90
3.1.8.2. La facturation	91
3.1.9. TP6 : Synthèse	92
3.1.9.1. La mise au point de la BDD	92
3.1.9.2. La partie administration du site	92

Chapitre 1. Les bases de PHP

Table des matières

1.1. Évaluation	2
1.1.1. TP1 : bases en PHP (exercices progressifs)	3
1.1.1.1. Objectifs et déroulement	3
1.1.1.2. Exercice 0 : configuration	3
1.1.1.3. Exercice 1	3
1.1.1.4. Exercice 2	4
1.1.1.5. Exercice 3	6
1.1.1.6. Exercice 4	8
1.1.1.7. Exercice 5	9
1.1.1.8. Exercice 6	9
1.1.2. TP2 : bases en PHP	13
1.1.2.1. Objectifs et déroulement	13
1.1.2.2. Exercice 1 : configuration	13
1.1.2.3. Exercice 2	13
1.1.2.4. Exercice 3	14
1.1.2.5. Exercice 4	14
1.1.2.6. Exercice 5	15
1.1.2.7. Exercice 6	15
1.1.3. TP3 : bases en PHP (tableaux & formulaires)	18
1.1.3.1. Objectifs et déroulement	18
1.1.3.2. Exercice 1 : tableau simple	18
1.1.3.3. Exercice 2 : différents type d'affichage	18
1.1.3.4. Exercice 3 : tableau une simple boucle	19
1.1.3.5. Exercice 4 : tableau entré par l'utilisateur	19
1.1.3.6. Exercice 5 : l'utilisateur sélectionne une entrée d'un tableau	19
1.1.3.7. Exercice 6 : manipulation de tableau	20
1.1.3.8. Exercice 7 : tableau multidimensionnel	21
1.1.3.9. Exercice 8 : tri à bulles	21

1.1. Évaluation

Voilà une batterie d'exercices, utilisez le manuel en ligne (i.e. **man**), le cours en ligne, vos documents de cours ...

Bon courage !



1.1.1. TP1 : bases en PHP (exercices progressifs)

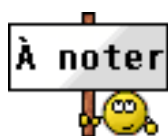
1.1.1.1. Objectifs et déroulement

Quelques exercices en PHP de préférence sous Linux.

- Pour ce TP vous travaillerez par groupes de deux (UN groupe de trois est permis si le nombre de personnes dans votre groupe de TP ne permet pas uniquement des groupes de deux) ;
- Vous aurez besoin d'une machine virtuelle **Debian** par étudiant.
- Dans le cadre de ce TP, vous remettrez une archive `tar.gz` contenant tous les codes PHP des exercices.

Barème approximatif. Questions Q1 à Q20 sur 1 point.

1.1.1.2. Exercice 0 : configuration



Sauf si vous travaillez sous Windows™ !

Dans ce cas vous devrez utiliser `easyphp` <http://www.easyphp.org/> [http://www.easyphp.org/] ! Cependant, même si vous travaillez sous Windows™ lisez ce qui suit car il est question de l'outil `phpmyadmin`.

Dans cet exercice vous devez vérifier la configuration des logiciels comme Apache, MySQL, PHP et activer le module `userdir` qui offre la possibilité au serveur Web de servir des pages situées dans le *home directory* de l'utilisateur courant. Les pages sont dans `/home/<compte>/public_html` et sont accessibles via `http://@ip_du_serveur/~<compte> ...`

Afin de gérer la BDD MySQL autrement qu'en ligne de commande l'outil `phpmyadmin` est tout indiqué. Pour faire installer par l'installateur de paquet Apache, PHP il suffit de lui demander d'installer `phpmyadmin`, il gère les dépendances. Cependant, il faut lui demander d'installer MySQL car `phpmyadmin` peut fonctionner sans qu'il y ai de BDD sur la machine. En d'autres termes `phpmyadmin` installé sur votre machine locale peut administrer éventuellement un serveur MySQL installé sur un autre ordinateur.

Voici la trame de ce qu'il faut faire.

```
[root@machina]# apt-get install phpmyadmin mysql-server
...
[user@machina]$ ls -l /etc/apache2/mods-enabled/❶
...
lrwxrwxrwx 1 root root 30 nov. 17 10:09 userdir.conf -> ../mods-available/userdir.conf
lrwxrwxrwx 1 root root 30 nov. 17 10:10 userdir.load -> ../mods-available/userdir.load
...
[user@machina]$
```

- ❶ Il faut vérifier que les liens symboliques sont bien créés dans le répertoire des modules d'Apache `mods-enabled` pointant sur le répertoire des modules eux-mêmes `mods-available`.

Enfin, il faut activer à l'aide de la commande `a2enmod` le module `userdir`.

1.1.1.3. Exercice 1

Coder ces trois exemples afin de vérifier leur fonctionnement.

Soit les codes sources PHP suivant. Les instructions du code PHP se placeront naturellement entre ces deux balises `<?php` et `>` ...

```
1
2 <html>
3 <head>
4   <title>Test PHP</title>
5 </head>
6 <body>
7   <?php echo '<p>Bonjour le monde</p>'; ?>
8 </body>
9 </html>
10
```

Ne vous inquiétez pas pour le moment, on décrira plus tard ce que produit cette ligne de PHP. Ce petit exemple est juste là pour vous montrer comment on insère du code PHP dans une page WEB. Comme tout bon langage de programmation, PHP offre la possibilité de commenter son code. Pour cela, deux techniques :

1. pour commenter une seule ligne de code PHP, on précédera cette ligne de deux slashes //
2. pour commenter une portion de code, on précédera la première ligne de code que l'on souhaite commenter par un /* et on fera suivre la dernière ligne de code que l'on souhaite commenter par un */

```
1
2 <html>
3 <head>
4   <title>Test phpinfo</title>
5 </head>
6 <body>
7   <?php phpinfo(); ?> </body>
8 </html>
9
```

Ce qui distingue le PHP des langages de script comme le Javascript est que le code est exécuté sur le serveur. Si vous avez un script similaire sur votre serveur, le client ne reçoit que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat. Vous pouvez configurer votre serveur web afin qu'il analyse tous vos fichiers HTML comme des fichiers PHP. Ainsi, il n'y a aucun moyen de distinguer les pages qui sont produites dynamiquement des pages statiques. Le grand avantage de PHP est qu'il est extrêmement simple pour les néophytes, mais offre des fonctionnalités avancées pour les experts. Ne craignez pas de lire la longue liste de fonctionnalités PHP. Vous pouvez vous plonger dans le code, et en quelques instants, écrire des scripts simples. Bien que le développement de PHP soit orienté vers la programmation pour les sites web, vous pouvez en faire bien d'autres usages.

```
1
2 <?php
3 // ceci est un commentaire sur une seule ligne
4
5 /* ceci est
6 un commentaire
7 sur plusieurs lignes */
8 ?>
9
```

1.1.1.4. Exercice 2

Étudions dans un premier cours la déclaration des différents types de variables. Pour simplifier les choses, nous allons admettre qu'une variable correspond à un espace de la mémoire où l'on peut stocker une information.

Or, afin de pouvoir récupérer cette information lorsque l'on en a besoin, nous allons attribuer un nom à notre variable (si mes prof m'entendaient parler, ils me zapperaient tous mes diplômes). En PHP, les variables sont représentées par une chaîne de caractères, ayant toujours comme premier caractère, le caractère dollar (\$). Les variables peuvent avoir n'importe quelle lettre en deuxième caractère du moment qu'il ne s'agit pas d'un chiffre.

De plus, on ne peut mettre d'espace dans le nom d'une variable. Puis, pour assigner une valeur à une variable, on tachera d'utiliser l'opérateur =, tout en prenant soin de toujours placer la variable qui reçoit le résultat d'une opération à gauche du signe =.

Démonstration.

1. Lorsque l'on désire affecter une chaîne de caractères à une variable, il faut placer cette chaîne de caractères entre deux ".
2. Lorsque l'on désire affecter une valeur numérique à une variable, il ne faut pas placer de " autour cette valeur (en fait, c'est possible de mettre des " autour d'une valeur numérique, mais ensuite, il faut être vraiment vigilant, car on pourrait faire la confusion entre une valeur numérique et une chaîne de caractères).

Voici quelques exemples de déclarations de variables.

Soit les codes sources PHP suivant.

```

1
2  <?php
3  $nom = "LE FORMATEUR";
4  // $nom contient alors la chaine de caracteres LE FORMATEUR
5
6  $mon_chiffre = 12;
7  // $mon_chiffre contient la valeur numerique 12.
8
9  $5toto = "test";
10 // Cette declaration n'est pas valide car le nom de la variable commence par un chiffre
11 ?>
12
13
```

Voyons maintenant la déclaration des variables de type tableau (array). Pour ceux qui sont débutants en programmation, nous allons prendre un exemple plutôt simple afin de comprendre ce qu'est un tableau. Imaginons un classeur d'élève (ce sera notre tableau) contenant différentes feuilles (qui seront les indices du tableau). Imaginons également que ces feuilles soient numérotées, et chaque feuille contienne un texte particulier. Dès lors, on peut chercher le contenu d'une feuille de ce classeur grâce à son numéro (on cherche donc l'information contenu dans le classeur à la page numéro x). En informatique, un tableau, c'est exactement la même chose que notre classeur. Il s'agit d'une variable contenant différentes informations (les textes) et ces informations sont classées suivant le numéro de l'indice (c'est à dire le numéro de la feuille). Par exemple, supposons que l'on ait la variable \$fruit de type array. On pourrait alors avoir le code suivant :

```

1
2  <?php
3  $fruit = Array();
4  $fruit[0] = "fraise";
5  $fruit[1] = "banane";
6  $fruit[2] = "abricot";
7  ?>
8
```

En reprenant l'exemple du classeur, c'est comme si nous avions un classeur de nom fruit, ayant 3 pages :

1. sur la page 0, on aurait l'information fraise
2. sur la page 1, on aurait l'information banane
3. sur la page 2, on aurait l'information abricot

Nous venons, dans ce bout de code, de déclarer une variable de type array qui comporte 3 éléments (les pages). Nous aurions eu le même résultat en exécutant le bout de code suivant :

```

1
```

```
2 <?php
3 $fruit = Array();
4 $fruit[] = "fraise";
5 $fruit[] = "banane";
6 $fruit[] = "abricot";
7 ?>
8
```

En revanche, cette syntaxe est moins lisible, vu que souvent, on n'arrive plus vraiment à savoir à quelle page se trouve l'information recherchée (on s'emmêle dans les indices). Aparté ! Au lieu d'utiliser des chiffres pour les indices (comme dans notre exemple où nous avons utilisé les indices 0, 1 et 2) nous pouvons très bien utiliser des chaînes de caractères. Ce qui pourrait alors donner :

```
1
2 <?php
3 $fruit = Array();
4 $fruit['le_meilleur'] = "fraise";
5 $fruit['le_prefere_de_Julien'] = "banane";
6 $fruit['mon_prefere'] = "abricot";
7 ?>
8
```

Or dans ce cas, il faut évidemment utiliser pour chaque indice du tableau, une chaîne de caractère unique. Nous pouvons également déclarer des tableaux à plusieurs éléments. Pour ceux qui désirent vraiment exploiter cette possibilité, je vous invite à aller consulter la documentation officielle PHP.

Pour les 2 derniers exercices vous afficherez les tableaux grâce à la fonction `print_r()`. Bien entendu, vous devrez faire une recherche dans la documentation PHP sur cette fonction.

1.1.1.5. Exercice 3

Lors de cet exercice, nous allons mettre en pratique notre premier bout de code en PHP. Il faut également savoir que toutes les variables en PHP commencent par le signe dollar (\$), et il faut également ne pas oublier de placer un ; à chaque fin d'instruction PHP.

Soit les codes sources PHP suivant.

```
1
2 <?php
3 $nom = "LE FORMATEUR";
4 echo 'Bonjour ' ;
5 echo $nom;
6 echo ' !';
7 ?>
8
9
```

En effet, nous avons placé la chaîne de caractères LE FORMATEUR dans la variable \$nom, puis nous avons demandé à PHP d'écrire la chaîne de caractères Bonjour, puis le contenu de la variable \$nom (qui contient donc la chaîne LE FORMATEUR), et enfin, nous affichons un !. Ce qui donne bien comme résultat "Bonjour LE FORMATEUR !" (Remarquons au passage qu'une variable contenant une chaîne de caractères se déclare en mettant le nom de la variable puis un signe = et enfin, la chaîne de caractères entre deux "). Au passage, remarquons également que nous venons d'apprendre notre première commande PHP, la commande echo(). Cette fonction nous permet d'afficher à l'écran des chaînes de caractères, qui peuvent être définies directement par l'utilisateur (comme lorsque l'on fait un echo 'Bonjour '); ou qui peuvent être des contenus de variables (lorsque l'on fait echo \$nom;).



Attention !!!

En effet, si nous avons écrit le code suivant :

```

1
2 <?php
3 $nom = "LE FORMATEUR";
4 echo 'Bonjour ';
5 echo '$nom';
6 echo ' !';
7 ?>
8
9

```

Nous aurions eu à l'écran : Bonjour \$nom ! Attention donc à bien utiliser les ' qui ne servent qu'à délimiter les chaînes de caractères, et donc, il ne faut surtout pas les utiliser pour afficher le contenu d'une variable. En effet, lorsque l'on tape echo '\$nom'; PHP pense que l'on désire afficher la chaîne de caractères \$nom et non le contenu de la variable \$nom. Prenons un autre exemple où nous allons afficher la date et l'heure du jour.

```

1
2 <?php
3 $date_du_jour = date ("d-m-Y");
4 $heure_courante = date ("H:i");
5 echo 'Nous sommes le : ';
6 echo $date_du_jour;
7 echo ' Et il est : ';
8 echo $heure_courante;
9 ?>
10

```

Ce qui affichera à l'écran : Nous sommes le 17-09-2002 Et il est 12:10

Les simples ou doubles côtes ne donnent pas le même résultat. Faites une recherche dans la documentation afin de bien assimiler la différence.

Concernant la fonction date :

Dans ce cas, nous venons d'utiliser la fonction date() qui nous permet d'afficher la date du jour ainsi que l'heure courante (en fait, la date du serveur). Dans un premier temps, nous avons affecté à la variable \$date_du_jour le contenu que retourne la fonction date() munie des paramètres "d-m-Y", soit 17-09-2002, puis nous avons affecté à la variable \$heure_courante le contenu que retourne la fonction date() munie des paramètres "H:i", soit 12:10.

Voici la liste des paramètres possibles pour la fonction date().

- a : "am" (matin) ou "pm" (après-midi)
- A : "AM" (matin) ou "PM" (après-midi)
- d : Jour du mois, sur deux chiffres (éventuellement avec un zéro) : "01" à "31"
- D : Jour de la semaine, en trois lettres (et en anglais) : par exemple "Fri" (pour Vendredi)
- F : Mois, textuel, version longue; en anglais, i.e. "January" (pour Janvier)
- h : Heure, au format 12h, "01" à "12"
- H : heure, au format 24h, "00" à "23"
- g : Heure, au format 12h sans les zéros initiaux, "1" à "12"
- G : Heure, au format 24h sans les zéros initiaux, "0" à "23"
- i : Minutes; "00" à "59"
- j : Jour du mois sans les zéros initiaux: "1" à "31"

- l : Jour de la semaine, textuel, version longue; en anglais, i.e. "Friday" (pour Vendredi)
- L : Booléen pour savoir si l'année est bissextile ("1") ou pas ("0")
- m : Mois; i.e. "01" à "12"
- n : Mois sans les zéros initiaux; i.e. "1" à "12"
- M : Mois, en trois lettres (et en anglais) : par exemple "Jan" (pour Janvier)
- s : Secondes; i.e. "00" à "59"
- S : Suffixe ordinal d'un nom

1.1.1.6. Exercice 4

Voyons maintenant les variables d'environnements.

En effet, PHP propose toute une série de variables qui sont déjà présentes dans le langage sans que vous n'ayez à les déclarer. Ces variables s'écrivent toujours en majuscules et nous fournissent divers renseignements.

`$_SERVER` est un tableau des variables fournies par le serveur web, ou bien directement liées à l'environnement d'exécution du script courant. C'est la nouvelle version de l'ancienne variable `$HTTP_SERVER_VARS`.

Le tableau `$_SERVER`

Variable	Utilité	Exemple d'affichage
<code>\$_SERVER['REQUEST_METHOD']</code>	La méthode d'appel	POST
<code>\$_SERVER['SERVER_NAME']</code>	Nom du serveur	localhost
<code>\$_SERVER['SERVER_ADMIN']</code>	L'email de l'administrateur du serveur	email@domaine.tld
<code>\$_SERVER['SERVER_ADDR']</code>	L'adresse IP du serveur	195.14.0.256
<code>\$_SERVER['QUERY_STRING']</code>	Les paramètres indiquées à votre script	url=toto.html&id=234
<code>\$_SERVER['REMOTE_PORT']</code>	Port de la requête	80
<code>\$_SERVER['REMOTE_ADDR']</code>	Adresse de l'internaute	88.101.2.255
<code>\$_SERVER['REQUEST_URI']</code>	Chemin du script	/exemple.php
<code>\$_SERVER['PATH_TRANSLATED']</code>	Chemin physique (complet) du script	/home/www/domain.fr/example.php
<code>\$_SERVER['HTTP_USER_AGENT']</code>	User agent du navigateur du client	Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1) Gecko/20061010 Firefox/2.0
<code>\$_SERVER['HTTP_REFERER']</code>	L'URL de la page d'où provient l'internaute	http://oseox.fr/exemple.php
<code>\$_SERVER['HTTP_HOST']</code>	Le nom de domaine où est exécuté le script	example.fr
<code>\$_SERVER['HTTP_ACCEPT_LANGUAGE']</code>	Langue acceptée par le navigateur de l'internaute	fr
<code>\$_SERVER['DOCUMENT_ROOT']</code>	Adresse de la racine du serveur	/var/www/exemple.fr/

Ces variables peuvent être utilisées n'importe quand dans vos scripts.

Voici un exemple où vous pouvez afficher l'adresse IP de la personne qui se connecte sur votre site :

```

2  <?php
3  echo 'Votre adresse IP est : ' . $_SERVER['REMOTE_ADDR'];
4  ?>
5

```

Ce qui affichera à l'écran : Votre adresse IP est : 80.12.45.26

1.1.1.7. Exercice 5

Voyons maintenant la concaténation de chaîne de caractères. Déjà, il faut bien savoir ce qu'est la concaténation de chaîne de caractères. Prenons un exemple simple avec deux chaînes de caractères :

1. la première chaîne de caractères sera : "J'apprend "

2. la seconde chaîne sera : "le PHP"

En faisant une concaténation de ces deux chaînes, nous obtiendrons la chaîne suivante : "J'apprend le PHP". Tachons maintenant de mettre alors en évidence l'importance de la concaténation de chaîne de caractères avec l'exemple de l'exercice précédent. On avait alors comme code PHP :

```

1
2  <?php
3  $nom = "LE FORMATEUR";
4  echo 'Bonjour ';
5  echo $nom;
6  echo ' !';
7  ?>
8
9

```

Or, les trois lignes avec l'instruction echo() peuvent se simplifier en une seule grâce à la concaténation. En PHP, la concaténation de chaîne s'effectue grâce au point. On a alors :

```

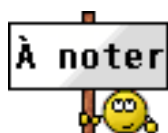
1
2  <?php
3  $nom = "LE FORMATEUR";
4  echo 'Bonjour ' . $nom . ' !';
5  ?>
6
7

```

Ce qui affichera à l'écran : Bonjour LE FORMATEUR !

Résumons les choses : On affiche en fait la chaîne Bonjour concaténée avec le contenu de la variable \$nom, soit LE FORMATEUR, également concatène avec la chaîne !, ce qui au final, se résume par l'affichage de la chaîne Bonjour LE FORMATEUR. La encore, il faut faire attention car si l'on avait écrit echo 'Bonjour \$nom !'; donc sans l'opérateur de concaténation qu'est le point, on aurait eu à l'écran l'affichage suivant :

Bonjour \$nom !



PHP comprend ...

Car dans ce cas, PHP comprend que l'on désire afficher la chaîne \$nom et non le contenu de la variable \$nom.

1.1.1.8. Exercice 6

Après avoir vu un premier aperçu du langage PHP, nous allons maintenant étudier les différentes structures de contrôles du langage. Les structures de contrôles nous permettront de faire des tests entre les variables et d'exécuter diverses boucles. Voici un petit récapitulatif des principales structures de contrôles.

Tableau 1.1. Structures de contrôles

Instruction	Signification
if	Si
else	Sinon
elseif	Sinon si
switch	Selon
for	Pour chaque (boucle)
while	Tant que (boucle)
==	Strictement égal
!=	Différent
<	Strictement inférieur
>	Strictement supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
and ou &&	ET logique
or ou	OU logique

Nous allons maintenant illustrer toutes ces structures de contrôles. Nous allons initialiser une variable numérique \$nombre à la valeur 11 par exemple, et faire différents tests dessus.

```
1
2 <?php
3 $nombre = 11;
4 if ($nombre >= 0 && $nombre < 10) {
5     // on teste si la valeur de notre variable est comprise entre 0 et 9
6     echo $nombre.' est compris entre 0 et 9';
7 }
8 elseif ($nombre >= 10 && $nombre < 20) {
9     // on teste si la valeur de notre variable est comprise entre 10 et 19
10    echo $nombre.' est compris entre 10 et 19';
11 }
12 else {
13     // si les deux tests précédents n'ont pas aboutis, alors on tombe dans ce cas
14     echo $nombre.' est plus grand que 19';
15 }
16 ?>
17
```

A l'affichage on aura : 11 est compris entre 10 et 19

Remarquons déjà que les instructions qui doivent être exécutées lorsqu'un test est validé sont systématiquement comprises entre des crochets { }. En effet, résumons ce qui vient de se passer. Dans un premier temps, on teste si \$nombre est supérieur ou égal à 0 et strictement inférieur à 10, et dans ce cas, et seulement dans ce cas, on écrira alors sur l'écran, \$nombre est compris entre 0 et 9. Or vu que \$nombre est égal à 11, on se rend compte que ce test ne sera pas satisfait. On a alors deux solutions. Soit on écrit directement le cas Sinon (else), soit on peut faire un autre test, ce qui correspond à un SinonSi (elseif). Nous avons choisi de faire un second test (elseif). Là, on teste si \$nombre est supérieur ou égal à 10 et strictement inférieur à 20 (ce qui est notre cas car \$nombre est égal à 11). Le test est donc validé, et l'on exécute alors les instructions présentes entre les crochets { } du elseif. On affichera donc à l'écran : 11 est compris entre 10 et 19. Viens ensuite le cas else qui est exécuté seulement si aucune des conditions définies par les if et les elseif n'est vérifiées.

Le switch représente exactement la même chose qu'une succession d'un if et de plusieurs elseif. En revanche, utiliser un switch à un certain avantage comparé à un if et à plusieurs elseif, c'est que sa structure est beaucoup moins lourde et nettement plus agréable à lire. Prenons un exemple simple. Nous allons déclarer une variable contenant une chaîne de caractères, puis nous allons tester cette chaîne grâce au switch.

On aura alors le code suivant :

```

1
2 <?php
3 $nom = "LE FORMATEUR";
4
5 switch ($nom) {
6     case 'Jean' :
7         echo 'Votre nom est Jean.';
8         break;
9     case 'Benoît' :
10        echo 'Votre nom est Benoît.';
11        break;
12    case 'LE FORMATEUR' :
13        echo 'Votre nom est LE FORMATEUR.';
14        break;
15    default :
16        echo 'Je ne sais pas qui vous êtes !!!';
17 }
18 ?>
19
20

```

Dans notre cas, vu que \$nom contient la chaîne de caractère LE FORMATEUR, on verra alors s'afficher à l'écran la phrase suivante :

Votre nom est LE FORMATEUR.

En revanche, si la variable \$nom avait contenu la chaîne de caractère "toto", ce même code aurait affiché à l'écran :

Je ne sais pas qui vous êtes !!!

En utilisant un if puis une succession de elseif, le code suivant aurait exactement eu le même affichage sur l'écran :

```

1
2 <?php
3 $nom = "LE FORMATEUR";
4
5 if ($nom == "Jean") {
6     echo 'Votre nom est Jean.';
7 }
8 elseif ($nom == "Benoît") {
9     echo 'Votre nom est Benoît.';
10 }
11 elseif ($nom == "LE FORMATEUR") {
12     echo 'Votre nom est LE FORMATEUR.';
13 }
14 else {
15     echo 'Je ne sais pas qui vous êtes !!!';
16 }
17 ?>
18
19

```

Attention ! Notez bien l'utilisation de break dans chaque cas de votre switch. Si celui-ci est omis, tous les messages s'afficheront.

for (pour chaque)

La structure de contrôle for nous permet d'écrire des boucles. En clair, cela veut dire que nous allons exécuter une série d'instructions un nombre de fois bien déterminé. Prenons l'exemple suivant :

```
1
2 <?php
3 $chiffre = 5;
4
5 // D  but de la boucle
6 for ($i=0; $i < $chiffre; $i++) {
7     echo 'Notre chiffre est diff  rent de '.$i.'<br />';
8 }
9 // Fin de la boucle
10
11 echo 'Notre chiffre est   gal    '.$i;
12 ?>
13
```

Ce qui affichera    l  cran :

Notre chiffre est diff  rent de 0
Notre chiffre est diff  rent de 1
Notre chiffre est diff  rent de 2
Notre chiffre est diff  rent de 3
Notre chiffre est diff  rent de 4
Notre chiffre est   gal    5

En effet, on initialise notre variable \$chiffre    5. On d  marre la boucle for qui dit que l  on va ex  cuter les instructions situ  es entre les crochets de la boucle ({ }) pour i variant de 0    \$chiffre-1 (donc jusqu   4), i   tant incr  menter    chaque passage de boucle (\$i++). (\$i varie de 0    \$chiffre-1 car on impose que \$i soit strictement inf  rieur    \$chiffre). On ex  cute alors 4 fois les instructions pr  sentes dans la boucle, et    chaque passage, \$i verra sa valeur augment  e de 1. Apart   : L'utilisation des boucles est extr  mement importante (et indispensable) en programmation. La compr  hension de ce passage est capital.

while (tant que)

Voyons maintenant l  autre boucle, la boucle while (dite boucle tant que). Il faut d  j   savoir que la boucle while n  est pas vraiment d  une n  cessit   absolue (elle est absente dans certains langages de programmation) vu qu  elle est toujours rempla  able par une boucle for. Reprenons l  exemple pr  c  dent, et   crivons le    l  aide de la boucle while, on a :

```
1
2 <?php
3 $chiffre = 5;
4 $i = 0;
5
6 // D  but de la boucle
7 while ($i < $chiffre) {
8     echo 'Notre chiffre est diff  rent de '.$i.'<br />';
9     $i = $i + 1;
10 }
11 // Fin de la boucle
12
13 echo 'Notre chiffre est   gal    '.$i;
14 ?>
15
```

Ce qui affichera    l  cran exactement la m  me chose que ce qu  affiche le code que l  on a utilis   pour la boucle for. Ici, on initialise notre variable \$chiffre    5, puis la variable \$i    0. Ensuite, nous faisons le test suivant : tant que \$i < \$chiffre, alors afficher "Notre chiffre est diff  rent de \$i" et augmenter la valeur de \$i de 1. Puis d  s que la condition \$i < \$chiffre n  est plus v  rifi  e, nous sortons de la boucle pour finir l  ex  cution des instructions qui suivent.

1.1.2. TP2 : bases en PHP

1.1.2.1. Objectifs et déroulement

Quelques exercices en PHP de préférence sous Linux.

- Pour ce TP vous travaillerez par groupes de deux (UN groupe de trois est permis si le nombre de personnes dans votre groupe de TP ne permet pas uniquement des groupes de deux) ;
- Vous aurez besoin d'une machine virtuelle **Debian** par étudiant.
- Dans le cadre de ce TP, vous remettrez une archive `tar.gz` contenant tous les codes PHP des exercices.

Barème approximatif. Questions Q1 à Q20 sur 1 point.

1.1.2.2. Exercice 1 : configuration

Dans cet exercice vous devez vérifier la configuration des logiciels comme Apache, MySQL, PHP et activer le module `userdir` qui offre la possibilité au serveur Web de servir des pages situées dans le *home directory* de l'utilisateur courant. Les pages sont dans `/home/<compte>/public_html` et sont accessibles via `http://@ip_du_serveur/~<compte>` ...

Afin de gérer la BDD MySQL autrement qu'en ligne de commande l'outil `phpmyadmin` est tout indiqué. Pour faire installer par l'installateur de paquet Apache, PHP il suffit de lui demander d'installer `phpmyadmin`, il gère les dépendances. Cependant, il faut lui demander d'installer MySQL car `phpmyadmin` peut fonctionner sans qu'il y ai de BDD sur la machine. En d'autres termes `phpmyadmin` installé sur votre machine locale peut administrer éventuellement un serveur MySQL installé sur un autre ordinateur.

Voici la trame de ce qu'il faut faire.

```
[root@machina]# apt-get install phpmyadmin mysql-server
...
[user@machina]$ ls -l /etc/apache2/mods-enabled/❶

...
lrwxrwxrwx 1 root root 30 nov. 17 10:09 userdir.conf -> ../mods-available/userdir.conf
lrwxrwxrwx 1 root root 30 nov. 17 10:10 userdir.load -> ../mods-available/userdir.load
...

[user@machina]$
```

- ❶ Il faut vérifier que les liens symboliques sont bien créés dans le répertoire des modules d'Apache `mods-enabled` pointant sur le répertoire des modules eux-mêmes `mods-available`.

Enfin, il faut activer à l'aide de la commande `a2enmod` le module `userdir`.

1.1.2.3. Exercice 2

Soit le code PHP suivant.

```
1
2 <html>
3 <head>
4 <title>Loops </title>
5 </head>
6 <body>
7
8 <?php
```

```
9
10 for ($intRed=0; $intRed<=255; $intRed=$intRed+30)
11     for ($intGreen=0; $intGreen<=255; $intGreen=$intGreen+30)
12         for ($intBlue=0; $intBlue<=255; $intBlue=$intBlue+30) {
13             $StrColor = "rgb(" . $intRed . "," . $intGreen . "," . $intBlue . ")";
14             echo "<span style='color:" . $StrColor . "'>" . $StrColor . "</span><br />";
15         }
16 ?>
17
18 </body>
19 </html>
20
```

Modifier le script suivant afin qu'il affiche des petits carrés de couleurs au lieu des `rgb(x,y,z)` ... Faites au moins deux exemples.

1.1.2.4. Exercice 3

Modifier le script précédent afin qu'il prenne un paramètre `typeaff` passé en GET permettant de choisir l'une ou l'autre des deux solutions proposées.

Pour avoir accès à cette page vous taperez l'une des adresses suivantes dans votre navigateur : **`http://@IP/~<compte>/exercice2.php?typeaff=1`** ou **`http://@IP/~<compte>/exercice2.php?typeaff=2`** ...

1.1.2.5. Exercice 4

La primitive `popen()` est bien utile car elle permet d'exécuter une commande et de récupérer son résultat dans le code qui l'a appelé.

Soit le code C suivant.

```
1
2 /* exemple_popen1.c */
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <string.h>
6 #include <unistd.h>
7 #include <errno.h>
8 #define NBCAR 128
9
10 int main (void)
11 {
12     FILE * sortie;
13     char ligne [NBCAR];
14     char etat [NBCAR];
15
16     if ((sortie = popen("/sbin/ifconfig eth0", "r")) == NULL) {
17         fprintf(stderr, "Erreur popen %d \n", errno);
18         exit(1);
19     }
20
21     while (fgets(ligne, NBCAR, sortie) != NULL) {
22         printf("La chaîne : %s", ligne);
23
24         /* %s dans sscanf() correspond à une séquence de
25          caractères différents des caractères blancs :
26          ici le 1er 'mot' de chaque ligne ! */
27         if (sscanf(ligne, "%s", etat) == 1)
28             /* printf("=>%s\n", etat); */
29             if (strcmp(etat, "UP") == 0) {
30                 printf("Etat : %s", etat);
31                 fprintf(stderr, "interface eth0 en marche\n");
32                 pclose(sortie);
33             }
34     }
35 }
```

```

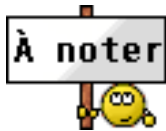
33     return 0;
34 }
35 }
36
37 fprintf(stdout, "interface eth0 inactive\n");
38 pclose(sortie);
39 return 0;
40 }
41
42
43

```

La primitive `popen()` existe aussi en PHP ! Réaliser un script PHP qui fait exactement la même chose que le code C ci-dessus.

Fonction `is_eth($n)` à développer

- Le script PHP comportera une fonction `is_eth($n)` qui prendra en paramètre le numéro de l'interface ;
- Cette fonction retournera `TRUE` ou `FALSE` suivant que l'interface est active ou non ;
- Vous appellerez la fonction et afficherez : L'interface `ethX` est : **active** ou **inactive** ;



L'opérateur *ternaire* !

Vous utiliserez avantageusement l'opérateur ternaire ! Pour l'affichage du résultat !

1.1.2.6. Exercice 5

Rajoutez au code PHP de l'exercice précédent la réactualisation de la page toutes les T secondes.

En deux étapes

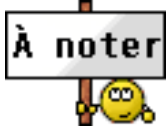
- Dans un premier temps vous coderez en dur dans le code une réactualisation toutes 5 secondes ;
- Puis vous paramètrerez cette temporisation via une variable passée en GET. Par exemple **`http://@IP/~<compte>/exercice5.php?T=10`** pour une réactualisation toutes les 10 secondes. Cependant, il faut penser au cas où l'argument n'est pas passé en GET et définir pour ce cas particulier une temporisation arbitraire de 5 secondes.

1.1.2.7. Exercice 6

Vous devez enregistrer dans une base de données les états de l'interface `ethX`. Ainsi que la date où l'état de celle-ci a été consultée.

Plusieurs étapes

- Il faut créer une BDD ;
- Il faut créer une table pour cette BDD ;
- Il vous faut créer une fonction de connexion à la BDD. Comportant des variables comme le login et le passord ainsi que l'adresse IP du serveur ;
- Il vous faudra créer également une fonction de déconnexion à la BDD ;
- Il vous faut une fonction d'insertion des valeurs dans une table ;
- Il vous faut une fonction de consultation des valeurs de la table ...



Gestion de la BDD

L'idée est d'utiliser phpmyadmin afin de simplifier la création de la BDD (ainsi qu'une table associée). Le code SQL pour la création de la base et de sa table est donné ci-dessous.

Création de la BDD.

```
1
2
3
4 CREATE DATABASE valsystem_db CHARACTER SET 'utf8';
5
6
7
```

Création de la table.

```
1
2
3
4 CREATE TABLE ethx (
5     id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
6     etat VARCHAR(20) NOT NULL,
7     date_du_jour DATETIME NOT NULL,
8     PRIMARY KEY (id)
9 )
10 ENGINE=MyISAM;
11
12
13
```

Connexion à la BDD.

```
1
2
3
4 $link connectDB($localhost,$login,$passwd,$bdd);
5
6
7
```

Deconnexion de la BDD.

```
1
2
3
4 closeDB($link);
5
6
7
```

Insertion des valeurs dans la table.

```
1
2
3
4 insertDB($link,$ethx,$val,$date);
5
6
7
```

Affichage des valeurs stockées en BDD.

```
1
2
3
4 affallinDB($link);
5
6
7
```

**C'est quoi \$link ?**

C'est ce que retourne `mysql_connect("localhost", "mysql_user", "mysql_password")` et qui permet ensuite de réaliser les requêtes sur la BDD. C'est en d'autres termes un identifiant de connexion.

1.1.3. TP3 : bases en PHP (tableaux & formulaires)

1.1.3.1. Objectifs et déroulement

Quelques exercices en PHP de préférence sous Linux.

- Pour ce TP vous travaillerez en autonomie ;
- Vous aurez besoin d'une machine virtuelle **Debian** par étudiant ;
- Dans le cadre de ce TP, vous remettrez une archive `tar.gz` contenant tous les codes PHP des exercices.

1.1.3.2. Exercice 1 : tableau simple

Soit le tableau PHP suivant.

```
1
2
3 <?php
4 //Create array.
5 $cities=array(
6     "Tokyo",
7     "Mexico",
8     "New York",
9     "Bombay",
10    "Séoul",
11    "Shanghai",
12    "Lagos",
13    "Buenos Aires",
14    "Caire",
15    "Londre"
16 );
17 ?>
18
19
```

Procédure 1.1. Vous devez coder le fichier `tp3_exo1.php`

1. Afficher les éléments du tableau dans une boucle `foreach()`, soit le nom des villes séparées par des virgules.
2. Trier les valeurs du tableau en utilisant `sort()` puis les afficher de nouveau.
 - a. D'une manière simple comme précédemment ;
 - b. D'une manière plus complexe dans des balises `<lu>ville 1ville 2</lu>` ;
3. Vous devez rajouter des valeurs ("Paris", "Los Angeles", "Calcutta") au tableau en utilisant `array_push()`. Puis faire l'affichage du tableau comme précédemment.

1.1.3.3. Exercice 2 : différents type d'affichage

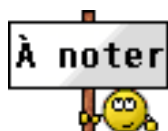
Procédure 1.2. Vous devez coder le fichier `tp3_exo2.php`

1. En reprenant le code PHP de l'exercice précédent changer le type d'affichage `lu/li` en utilisant une structure de type tableau.
2. Vous coderez une fonction `aff($tab,$typeaff)` permettant de choisir le type d'affichage `lu/li` ou `table` suivant la valeur d'un paramètre `$typeaff` prenant comme valeur soit `lilu` ou `table`. L'argument `$tab` étant le tableau à afficher.

1.1.3.4. Exercice 3 : tableau une simple boucle

Pour cet exercice de PHP, créez un formulaire demandant à l'utilisateur d'entrer le climat que l'utilisateur a connu pour une période et un lieu donné. Dans des champs de texte séparés, demandez à l'utilisateur d'entrer la ville, le mois et l'année en question. En dessous, vous montrerez une série de cases à cocher indiquant les conditions météorologiques. Ces valeurs sont : la pluie, le soleil, les nuages, la grêle, le verglas, la neige, le vent, le froid, la chaleur ...). Mettre en place le formulaire afin de créer un tableau `name="temps[]"` à partir des éléments vérifiés.

Dans la section de votre script de réponse (qui sera sur la même page), créer un tableau en utilisant la ville, mois et l'année que l'utilisateur a entré. Afficher la réponse suivante : "Dans la ville `$inputLocal[0]` au mois de `$inputLocal[1]` en `$inputLocal[2]`, vous avez observé le temps qui suit ... L'affichage des résultats étant ceux que l'utilisateur a sélectionné sous forme d'une liste à puce après validation du formulaire.



`$inputLocal[0]`, `$inputLocal[1]` et `$inputLocal[2]` ?

Sont des valeurs provenant du formulaire passé en POST ! Ces valeurs sont donc tout simplement affectées aux variables `$POST['valeur']` ...

1.1.3.5. Exercice 4 : tableau entré par l'utilisateur

Dans cet exercice de PHP, vous allez créer un tableau, l'afficher dans le navigateur, puis demander à l'utilisateur d'y ajouter ses informations.

Créer un tableau de modes de transport, comprenant l'automobile, Jet, Ferry, Sous-marin, Métro. Afficher la déclaration suivante dans le navigateur : *Le voyage prend de nombreuses formes, que ce soit à travers la ville, à travers le pays, ou dans le monde entier. Voici une liste de quelques modes communs de transport : ...*

Lorsque l'utilisateur clique sur *Valider*, traiter l'entrée avec des fonctions de tableau et renvoyer la liste originale avec les ajouts de l'utilisateur. Inclure une autre zone de texte : *Voulez vous en ajouter ?* et un autre bouton d'envoi (*Valider*).

Lorsque l'utilisateur clique sur ce bouton, la page doit se recharger avec les nouveaux ajouts ajoutés à la liste. Votre code devrait permettre à l'utilisateur d'ajouter des éléments autant de fois qu'ils le souhaitent.



Cet exercice demande d'utiliser certaines fonctions et principes ...

Le tableau des moyens de transport est censé changer au fil des *submit* du formulaire, et pour cela il faudra rajouter en champ caché, afin qu'il ne reparte pas avec l'initialisation de départ les éléments (boucle) : `<input type="hidden" name="voyage[]" value="Automobile" />` `<input type="hidden" name="voyage[]" value="Avion" />` `<input type="hidden" name="voyage[]" value="Jet" />` ...

Vous devez utiliser des fonctions propres aux tableaux PHP sinon cet exercice risque d'être impossible à coder : `array_splice(...)` ou `array_merge(...)` ...

Une fonction qui permet d'enlever les espaces des variables afin de les *nettoyer* : `trim(...)`.

1.1.3.6. Exercice 5 : l'utilisateur sélectionne une entrée d'un tableau

Dans cet exercice de PHP, vous devez de nouveau utiliser la liste des villes de l'exercice 2. Voici la liste, incluant cette fois les pays et les villes :

Pays / ville

- Japon => Tokyo ;
- Mexique => La ville de Mexique ;
- USA => La ville de New York ;
- Inde => Bombay ;
- Corée => Séoul ;
- Chine => Shanghai ;
- Nigéria => Lagos ;
- Argentine => Buenos Aires ;
- Egypte => Caire ;
- Royaume-Uni => Londres.

Cette fois, créez un tableau associatif, en utilisant les pays comme clés et les villes comme valeurs.

Créer un formulaire pour l'utilisateur, avec les instructions ... *S'il vous plaît choisissez une ville :*

Faites des champs select / option pour les 10 villes, en bouclant sur le tableau.

1.1.3.7. Exercice 6 : manipulation de tableau

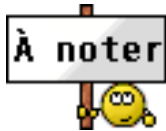
Dans cet exercice de PHP, vous allez créer un panel de températures. Choisissez le mois de printemps pour avoir une plus large gamme de températures à manipuler (on est en métropole pour cet exercice). Nous allons utiliser les 30 jours du mois.

Soit le tableau du tableau PHP suivant.

```
1
2
3 <?php
4 $temperatures = array(
5
6 20.10,22.50,23.30,15.00,16.50,27.80,28.00,23.50,24.00,25.00,23.50,15.00,18.90,27.00,26.50,
7 20.00,23.00,23.50,27.00,27.20,20.00,23.50,24.00,25.00,23.50,26.50,28.00,30.00,32.00,29.50
8 );
9
```

Procédure 1.3. Vous devez coder le fichier tp3_exo6.php

1. Afficher la température moyenne en degré.
2. Afficher les cinq températures les plus élevées.
3. Afficher les cinq températures les plus basses.



Comment faire avec le tableau ?

Il faut classer puis retirer 5 valeurs du tableau. Soit au début soit à la fin du tableau !

Utilisation de `rsort($temperatures);` et de `array_slice(...);` ...

1.1.3.8. Exercice 7 : tableau multidimensionnel

Dans cet exercice PHP, écrire le grandes villes tableau dans un tableau multi-dimensionnel et l'afficher.

Soit le tableau du tableau PHP suivant.

```

1
2
3 <?php
4 $multiVilles=array(
5     array('Ville', 'Pays', 'Continent'),
6     array('Tokyo', 'Japon', 'Asie'),
7     array('Ville de Mexique','Mexique', 'Amérique du Nord'),
8     array('La ville de New York', 'USA', 'Amérique du Nord'),
9     array('Bombay', 'Inde', 'Asie'),
10    array('Séoul', 'Corée', 'Asie'),
11    array('Shanghai', 'Chine', 'Asie'),
12    array('Lagos', 'Nigeria', 'Afrique'),
13    array('Buenos Aires', 'Argentine', 'Amérique du sud'),
14    array('Caire', 'Egypte', 'Afrique'),
15    array('Londre', 'Royaume-Uni', 'Europe')
16 );
17 ?>
18
19
```

1.1.3.9. Exercice 8 : tri à bulles

En vous inspirant des algorithmes du tri à bulles codez ceux ci en PHP.

Soit les algorithmes suivants.

```

1
2
3 fonction Tri_a_bulle(tableau)
4 DEBUT
5     taille = taille_du_tableau(tableau)
6
7     POUR I=0 a I<taille FAIRE
8         POUR J=0 a J<taille-1-I FAIRE
9             SI (tableau[J+1] < tableau[J])
10    Echange(tableau,J,I+1)
11        FINSI
12    FINFAIRE
13    FINFAIRE
14 FIN
15
16 fonction Echange(tableau,a,b)
17 DEBUT
18     TMP=tableau[a]
19     tableau[a] = tableau[b]
20     tableau[b] = TMP
21 FIN
22
23 TABLEAU = (1,3,2,8,5,7,4,0)
```

```
24 NEWTAB = Tri_a_bulle(TABLEAU)
25
26 AFFICHER_TABLEAU(NEWTAB)
27
28
```

Chapitre 2. Sessions et cookies

Table des matières

2.1. Cours	25
2.1.1. À propos	26
2.1.1.1. Objectifs	26
2.1.1.2. Licence	26
2.1.2. Cookies	27
2.1.2.1. Généralités	27
2.1.2.1.1. Définition	27
2.1.2.1.2. Stockage	27
2.1.2.1.3. Caractéristiques	27
2.1.2.1.4. Les différents champs	27
2.1.2.1.5. Exemple fichier cookies	28
2.1.2.2. Écriture d'un cookie	29
2.1.2.3. Lecture d'un cookie	30
2.1.2.4. Effacement d'un cookie	31
2.1.3. Sessions	32
2.1.3.1. Définition	32
2.1.3.2. Initialisation	32
2.1.3.3. Les fonctions de gestion	32
2.1.3.3.1. <code>session_start()</code>	33
2.1.3.3.2. <code>session_register(nom1, nom2, ...)</code>	33
2.1.3.3.3. <code>session_destroy()</code>	33
2.1.3.3.4. <code>session_unset()</code>	33
2.1.3.4. Les fonctions d'accès aux variables	34
2.1.3.4.1. <code>session_is_registered()</code>	34
2.1.3.4.2. <code>session_unregister()</code>	34
2.1.3.4.3. <code>session_name([opt nom])</code>	34
2.1.3.4.4. <code>session_id([opt id])</code>	35
2.1.3.5. Exemples d'implémentation	35
2.1.3.5.1. Création d'un espace membre	35
2.1.3.5.2. Compteur de visiteurs online	36
2.1.3.6. Sérialisation des données	36
2.2. Évaluation	38
2.2.1. TP1 Série PHP	39
2.2.1.1. Exercice 1	39
2.2.1.2. Exercice 2	39
2.2.1.3. Exercice 3	40
2.2.1.4. Exercice 4	40
2.2.1.5. Exercice 5	40
2.2.1.6. Exercice 6	41
2.2.1.7. Exercice 7	41
2.2.2. TP2 les Cookies	42
2.2.2.1. Exercice 1	42
2.2.2.2. Exercice 2	42
2.2.2.3. Exercice 3	42
2.2.2.4. Exercice 4	42
2.2.3. TP3 les Sessions	44
2.2.3.1. Exercice 1	44
2.2.3.2. Exercice 2	44
2.2.3.3. Exercice 3	44
2.2.3.4. Exercice 4	44
2.2.4. TP4 synthèse	46
2.2.4.1. Généralités	46

2.2.4.2. Exercice 1 : les statistiques d'un site	46
2.2.4.2.1. Sujet	46
2.2.4.2.2. Etude du problème	47
2.2.4.2.3. Méthode de test	47
2.2.4.2.4. Affichage des statistiques	47
2.2.4.3. Exercice 2 : un logiciel de chat	47
2.2.4.3.1. Sujet	47
2.2.4.3.2. Etude du problème	47
2.2.4.3.3. Améliorations	47
2.2.5. TP5 : Gestion d'un caddie en PHP	48
2.2.5.1. Objectifs	48
2.2.5.2. L'art des choix	48
2.2.5.3. Utilisation des sessions	48
2.2.5.3.1. Définition des actions	49

2.1. Cours

2.1.1. À propos

2.1.1.1. Objectifs

Permettre la maîtrise des deux implémentations d'authentification **cookies** et **sessions**.

2.1.1.2. Licence

Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre **GNU** (**GNU** Free Documentation License), version 1.1 ou toute version ultérieure publiée par la Free Software Foundation ; sans Sections Invariables ; sans Textes de Première de Couverture ; et sans Textes de Quatrième de Couverture.

Une copie de la présente Licence est disponible ici [<http://www.fsf.org/copyleft/gpl.html>].

2.1.2. Cookies

Les cookies permettent de conserver un état au cours de transactions effectuées avec le protocole de communication HTTP. En effet, le protocole HTTP est dit **sans état** (*stateless*) lors d'une nouvelle transaction de données entre le client et le serveur, le serveur ne *se souvient pas* du précédent passage de l'internaute d'où l'idée de garder les traces de passage dans un petit fichier stocké sur l'hôte client. Les cookies ont été inventés par Lou Montulli alors qu'il travaillait chez Netscape Communications™, dans le but d'adapter dynamiquement le contenu des sites Web aux habitudes de navigation de l'internaute .

2.1.2.1. Généralités

2.1.2.1.1. Définition

La définition du dictionnaire est simple : *Un biscuit (cookie en anglais) est une pâtisserie cuite au four, composée d'un mélange de farine, d'oeufs et de sucre.* Mais en ce qui nous concerne² ce sera : *Un cookie est un fichier texte de taille limitée (<=4ko) permettant de stocker certaines informations.* Le but premier de ce type de fichier est de pouvoir garder en mémoire les informations d'un visiteur afin de pouvoir les réutiliser à chacune de ses visites. Par exemple on y stocke son NOM, son PRENOM, son Nickname, etc

2.1.2.1.2. Stockage

Suivant les systèmes d'exploitation et les différents navigateurs les cookies ne seront pas stockés aux mêmes endroits. En ce qui concerne MSIE™ les cookies sont stockés dans le répertoire C:\Windows\Cookies pour Windows™ 9x, C:\Documents and Setting\votrelogin\Local Settings\Temporary Internet Files\Cookies pour Windows™ XP (il vous faut activer l'option permettant de visualiser les fichiers cachés), enfin pour Netscape™ 7 ils sont stockés dans C:\Documents and Settings\votrelogin\Application Data\Mozilla\Profiles\votreprofil\cookies.txt. Sous **Linux** ils sont stockés dans le répertoire personnel de l'utilisateur \$HOME/.netscape/ ou \$HOME/.mozilla/ dans un fichier cookies.txt (**Mozilla** est la version libre de Netscape™ ...).

2.1.2.1.3. Caractéristiques

Quelques caractéristiques importantes relatives aux cookies :

- La taille d'un cookie (son nom + sa valeur) ne peut dépasser **4 ko** (la valeur étant automatiquement tronquée si dépassement) ;
- Le navigateur accepte au maximum **20 cookies** par serveur (en fait par domaine, selon la définition ci-dessous) ;
- Le navigateur ne mémorise au total que **300 cookies** (les cookies les plus anciens étant automatiquement détruits quand de nouveaux sont créés par un serveur) ;
- Un serveur ne peut par défaut manipuler (lire, mettre à jour, effacer) que les cookies qu'il a lui-même créés (*sous réserve de trous de sécurité du logiciel navigateur* !).

Une spécification sur les cookies se trouve sur le site de Netscape™ **Persistent Client State HTTP Cookies** [http://wp.netscape.com/newsref/std/cookie_spec.html], ne pas oublier la **RFC 2109** [<http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc2109.html>].

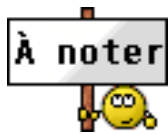
2.1.2.1.4. Les différents champs

¹
² Les différents champs écrits sur la machine cliente.

c.f. **Wikipedia** [<http://fr.wikipedia.org/wiki/Accueil>]

Bien sûr cela n'est possible qu'à partir du moment où le visiteur a entré lui-même ses informations dans un formulaire sur le site c.f. **PHPdébutant** [<http://www.phpdebutant.org/article40.php>]

domaine	Nom du serveur et de domaine Internet (serveur.domaine.net) pour lesquels l'accès au cookie est validé. Ce champ peut contenir un nom incomplet de type .domaine.net (avec au minimum 2 points), ce qui validera l'accès au cookie à tous les serveurs du domaine spécifié.
flag	TRUE/FALSE : Indique si toutes les machines dans un domaine donné peuvent accéder à la variable. Cette valeur est placée automatiquement par le navigateur, selon la valeur qui est placée pour le domaine.
path	Chemin de l'URL (partie qui suit http://serveur.domaine.net) définissant l'arborescence des ressources, sur le serveur, autorisées à accéder au cookie. Si l'on définit simplement /, toutes les <i>pages</i> du serveur.domaine.net pourront alors accéder à ce cookie. Si l'on définit un path sans spécifier de <i>page</i> Web, toutes les <i>pages</i> se trouvant dans le répertoire correspondant et ses sous-répertoires pourront alors accéder au cookie. Si l'on précise le path en le terminant par un nom de <i>page</i> Web, alors seule cette <i>page</i> pourra accéder au cookie. Par <i>page</i> il faut entendre bien évidemment tout type de ressource Web (page HTML, page PHP, script CGI...).
sécurité	TRUE/FALSE : Valeur logique indiquant s'il est nécessaire, respectivement pas nécessaire, de réaliser une transaction sécurisée (SSL) pour accéder au cookie.
expiration	Date et heure d'expiration du cookie. Au niveau de la transaction HTTP, cette date/heure est véhiculée en temps absolu au format GMT (p.ex. Sat 23-Mar-02 12:30:55 GMT), mais elle est stockée différemment dans les fichiers de cookies. Au-delà de cette date, le cookie sera automatiquement éliminé du fichier de cookies par le navigateur. Lorsqu'un cookie est mis en place sans que ne soit spécifiée une date d'expiration, celui-ci sera temporaire, c'est-à-dire qu'il disparaîtra automatiquement lorsque l'on quittera le navigateur (c-à-d. ne sera pas enregistré dans le fichier de cookies).
nom	Nom du cookie.
valeur	Valeur (chaîne de caractère) associée au nom du cookie. Au niveau de la transaction HTTP et dans le fichier de cookies, les caractères spéciaux (en particulier "point-virgule", "virgule", "espace", "tab") ne peuvent pas être utilisés tels quels .



Timestamp Unix™

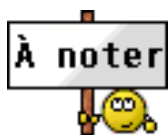
er

La date d'expiration, dans le fichier de cookies de Netscape™ et **Mozilla**, est exprimée en nombre de secondes écoulées depuis le 1 janvier 1970.

2.1.2.1.5. Exemple fichier cookies

Le fichier cookies.txt est issu d'une station de travail **Linux**.

```
[user@machinal]$ cat .mozilla/bruno/n4exo1o2.slt/cookies.txt | grep linuxfr.org
linuxfr.org FALSE / FALSE 1122474127 unique_id
w2d490UplgYrI2KIJ56Iq5tcnnczZELa
linuxfr.org FALSE /test FALSE 1122474127 md5
1d804064b0cfeebbb0d204831a48949f
linuxfr.org FALSE / FALSE 1122474127 login bguegan
```



Une seule machine visible de l'internet

Ici le second champ est à **FALSE**. Ceci est dû au fait qu'il y ait une seule machine pour ce domaine disponible de l'internet, cette machine est **www**, le domaine est **linuxfr**. Ici la notation **www.linuxfr.org** pour le premier champ serait identique.

3

Raison pour laquelle les fonctions de manipulation de cookies encodent/décodent les caractères dits spéciaux. Dans le cas du JavaScript voir les fonctions `escape()` et `unescape()` (remplacement du caractère par son code ASCII hexadécimal, p.ex. espace devient %20)

2.1.2.2. Écriture d'un cookie

Les cookies étant véhiculés sous forme d'en-têtes HTTP, la mise en place de cookies doit s'effectuer durant la phase d'envoi au navigateur des lignes d'en-têtes, donc avant la génération du corps de la réponse ! Cela vous impose d'appeler cette fonction avant toute balise <html> ou <head>. Si quelque chose a été envoyé avant l'appel à cette fonction, `setcookie` échouera et retournera **FALSE**. Si `setcookie` réussit, elle retournera **TRUE**. Cela n'indique pas si le client accepte ou pas le cookie.

```
bool setcookie(string name, string value, int expire, string path, string domain,
int secure);
```

Tableau 2.1. Paramètres de `setcookie`

Paramètre	Description	Exemples
name	Le nom du cookie.	'cookienom' est appelé via <code>\$_COOKIE['cookienom']</code>
value	La valeur du cookie. Cette valeur est stockée sur l'ordinateur du client ; ne pas stocker d'informations importantes.	Le paramètre name est le 'cookienom', cette valeur est retrouvée en utilisant <code>\$_COOKIE['cookienom']</code>
expire	Le temps après lequel le cookie expire. C'est un timestamp Unix™, donc, ce sera un nombre de secondes depuis l'époque Unix™ (1 Janvier 1970). En d'autres mots, vous devriez fixer cette valeur à l'aide de la fonction <code>time()</code> et en y ajoutant le nombre de secondes après lequel on veut que le cookie expire. Vous pouvez utiliser aussi <code>mktime()</code> .	<code>time()+60*60*24*30</code> fera expirer le cookie dans 30 jours. Si vous ne spécifiez pas ce paramètre, le cookie expirera à la fin de la session (lorsque le navigateur sera fermé). Vous pouvez également utiliser la fonction <code>mktime()</code> permettant de définir une date exacte au lieu d'une période de validité.
path	Le chemin sur le serveur sur lequel le cookie sera disponible.	Si la valeur est '/' , le cookie sera disponible sur l'ensemble du domaine domain. Si la valeur est '/foo/' , le cookie sera uniquement disponible dans le répertoire /foo/ ainsi que tous ces sous-répertoires comme /foo/bar/ du domaine domain . La valeur par défaut est le répertoire courant où le cookie a été défini.
domain	Le domaine où le cookie est disponible.	Pour rendre le cookie disponible sur tous les sous-domaines de example.com, vous devez mettre la valeur 'example.com' . Le point (.) n'est pas requis mais est nécessaire pour la compatibilité avec encore plus de navigateurs. Positionnez le à www.example.com rendra le cookie disponible uniquement sur le sous-domaine www . Reportez-vous aux spécifications pour plus de détails.
secure	Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS. Lorsqu'il est positionné à TRUE , le cookie ne sera positionné uniquement si la connexion sécurisée existe. La valeur par défaut est FALSE .	0 ou 1

Une fois que le cookie a été placé, il est accessible dans les variables globales `$_COOKIE` ou bien `$HTTP_COOKIE_VARS` arrays. Les valeurs de cookies existent aussi dans la variable `$_REQUEST`.

```
1
2
3 $expire = 365*24*3600; // on définit la durée du cookie, 1 an
4 setcookie("nickname","pascal",time()+$expire); // on l'envoie
5
6
```

```
1
2
3 // Avec mktime()
4 $expire=mktime(0, 0, 0, 12, 1, 2005); // A minuit le 01.12.2005
5 $expire=mktime(15, 50, 25, 12, 4, 2005); // A 15:50:25 le 04.12.2005
6 setcookie("nickname","pascal",$expire);
7
8
```

Vous pouvez aussi utiliser les cookies avec des tableaux, en utilisant la notation des tableaux. Cela a pour effet de créer autant de cookies que votre tableau a d'éléments, mais lorsque les cookies seront reçus par votre script, les valeurs seront placées dans un tableau :

```
1
2
3 setcookie("cookie[three]", "cookiethree" );
4 setcookie("cookie[two]", "cookietwo" );
5 setcookie("cookie[one]", "cookieone" );
6
7 // Après avoir rechargé la page :
8 if (isset($_COOKIE['cookie'])) {
9     foreach ($_COOKIE['cookie'] as $name => $value) {
10         echo "$name : $value <br />\n";
11     }
12 }
13
14
```



Comment accéder à la valeur d'un cookie ?

Ici, il vous faut recharger la page. En effet, les cookies ne sont accessibles qu'au chargement de la prochaine page, ou au rechargement de la page courante.

three : cookiethree

two : cookietwo

one : cookieone

2.1.2.3. Lecture d'un cookie

La lecture du cookie s'effectue exactement de la même manière qu'un paramètre transmis par l'url ...

```
1
2
3 // Afficher un cookie
4 echo $_COOKIE["nickname"];
5 echo $_HTTP_COOKIE_VARS["nickname"];
6
7 // Une autre méthode pour afficher tous les cookies
8 print_r($_COOKIE);
9
10
```

2.1.2.4. Effacement d'un cookie

L'effacement est très simple, il suffit d'appeler la fonction `setcookie` avec comme argument le nom du cookie, pour valeur le champ *vide* et tous les autres champs renseignés exactement comme lors de la création du cookie.

```
setcookie ("nickname", "");
```

Lorsque vous effacez un cookie, vous devriez toujours vous assurer que sa date d'expiration est déjà passée, pour déclencher le mécanisme de votre navigateur. Voici comment procéder :

```
1
2
3 // utilisation de la date moins une heure
4 setcookie ("nickname", "", time() - 3600);
5 setcookie ("nickname", "", time() - 3600, "~/rasmus/", ".example.com", 1);
6
7
```

Sinon, si la date d'expiration n'était pas dépassée, il n'y aurait pas d'effacement. En fait, ceci permet de déclencher le mécanisme qui vérifie que la date d'expiration est déjà passée.

2.1.3. Sessions

Combien d'entre vous se sont déjà demandés comment passer des infos pour un visiteur donné de page en page ? Et qui plus est, de manière sécurisée ? Vous pouvez imaginer, si jamais vous avez décidé un jour de mettre en place un espace membre avec des fonctionnalités avancées, qu'il est nécessaire d'identifier ce dit membre.

2.1.3.1. Définition

Une session est un mécanisme assez simple géré de manière transparente par le PHP. A chaque fois qu'une session est démarrée (3 méthodes sont possibles, comme nous le verrons plus tard), un identifiant unique est attribué au visiteur. Cet identifiant permettra de le distinguer des autres utilisateurs tout au long de sa session. Pendant la session, on pourra à loisir définir des variables qui *suivront le visiteur*.

A chaque chargement de page, ces variables seront sauvegardées sous forme sérialisée dans un fichier texte portant le nom de la session. Ces fichiers sont, en général stockés dans /tmp sur un système Unix™.

cookie Un cookie est posé au début de session et contient la variable PHPSESSID faisant référence à l'identifiant de session.

URL Le paramètre de session PHPSESSID ou SID est à inclure sur chaque lien de la page afin de propager la session aux autres pages.

La première solution est contraignante si vos utilisateurs refusent les cookies, mais reste tout de même la plus simple à mettre en place. A savoir tout de même que si PHP est compilé avec `--enable-trans-sid`, les urls de la page seront modifiées par PHP pour rajouter l'identifiant de session.



Où activer la session ?

A noter également que pour une création de cookie de session, l'appel de la fonction doit se situer avant toute autre sortie vers le navigateur, comme les fonctions `header()` et `setcookie()`.

2.1.3.2. Initialisation

Avant d'utiliser une session, il faut d'abord l'initialiser.

PHP est configuré pour démarrer automatiquement les sessions

PHP peut être configuré de telle façon, via le fichier de configuration `php.ini`, qu'il démarrera une session à chaque nouveau visiteur du site. Pour activer cette option, il suffit de passer le paramètre de configuration `session.auto_start` à 1 dans la configuration. C'est la façon la plus simple, mais pas forcément la plus répandue chez les hébergeurs.

Démarrer explicitement une session

PHP possède la fonction `session_start()`, qui, comme son nom l'indique, permet de démarrer une session. Si le visiteur affiche pour la première fois une page de votre site depuis qu'il a lancé son navigateur, un identifiant lui est attribué.

Autrement, s'il navigue depuis quelques pages, l'identifiant de session est récupéré et les variables lui étant associées sont accessibles.

Démarrer implicitement une session

On peut se passer du démarrage de session avec `session_start()` si l'on utilise la fonction `session_register()`, qui permet de stocker une variable dans une session.

Par cette méthode, PHP tente de déterminer si une session est active, et si ce n'est pas le cas, il la démarre avant de stocker la variable.

2.1.3.3. Les fonctions de gestion

2.1.3.3.1. session_start()

Comme nous l'avons vu précédemment, cette fonction, qui ne prend pas d'argument, permet d'initialiser une session, ou de continuer une session déjà existante.

2.1.3.3.2. session_register(nom1, nom2, ...)

Cette fonction permet de stocker une variable dans une session. Son emploi demeure simple, il suffit de passer le nom de la variable à stocker en tant qu'argument.

```
1
2
3 session_start();
4 $login = 'jesus';
5 $password = '1234';
6 session_register('login', 'password');
7
8
```

Cet exemple stockera les variables \$login et \$password dans la session, et par cette méthode, elles seront accessibles pour les autres pages affichées durant cette session. A noter cependant que l'on ne peut passer à la fonction session_register que des variables globales, et non locales.

Le code ci-dessous ne fonctionnera pas car la variable \$variable est locale :

```
1
2
3 function test_session() {
4     $variable = '1234';
5     session_register('variable');
6 }
7 test_session();
8 echo "var de session : {$_SESSION['variable']}";
9
10
```

Ici la variable est assignée aux variables globales par l'instruction \$GLOBALS['variable'] = '1234';, et corrige le problème :

```
1
2
3 function test_session() {
4     $GLOBALS['variable'] = '1234';
5     session_register('variable');
6 }
7
8 test_session();
9 echo "var de session : {$_SESSION['variable']}";
10
11
```

2.1.3.3.3. session_destroy()

Si l'on peut créer une session, il faut pouvoir la détruire, session_destroy() le permet, il suffit de l'appeler et la session en cours sera supprimée.

Il faut tout de même remarquer que les variables globales apportées par cette session avant sa destruction ne sont pas supprimées. Pour cela, il faut utiliser la fonction session_unset().

2.1.3.3.4. session_unset()

Cette fonction détruit les variables apportées par la session. Pas d'argument, juste un simple appel.

2.1.3.4. Les fonctions d'accès aux variables

Vous avez à votre disposition 2 méthodes pour manipuler les variables de session.

Vous pouvez utiliser directement le nom de la variable que vous avez passée à la fonction `session_register()`. Mais vous pouvez aussi y accéder via le tableau `$_SESSION['nom_de_variable']`.

De même, plutôt que d'utiliser `session_register()` pour incorporer une variable dans la session, il est également possible de la définir via `$_SESSION['nom']`.

Toutefois, si vous décidez d'utiliser cette méthode, il ne faudra plus utiliser les fonctions `session_register()`, `session_is_registered()` ou `session_unregister()`.

2.1.3.4.1. `session_is_registered()`

Cette fonction, vous l'aurez compris, permet de tester si une variable fait partie d'une session. Si, comme précisé précédemment, vous avez utilisé `$_SESSION`, il vous suffira de faire :

```
isset($_SESSION['ma_variable']);
```

2.1.3.4.2. `session_unregister()`

Dans la même série, cette fonction permet de détruire une variable de session. Son équivalent est `unset()`.

```
session_unregister('ma_variable');
```

```
unset($_SESSION['ma_variable']);
```

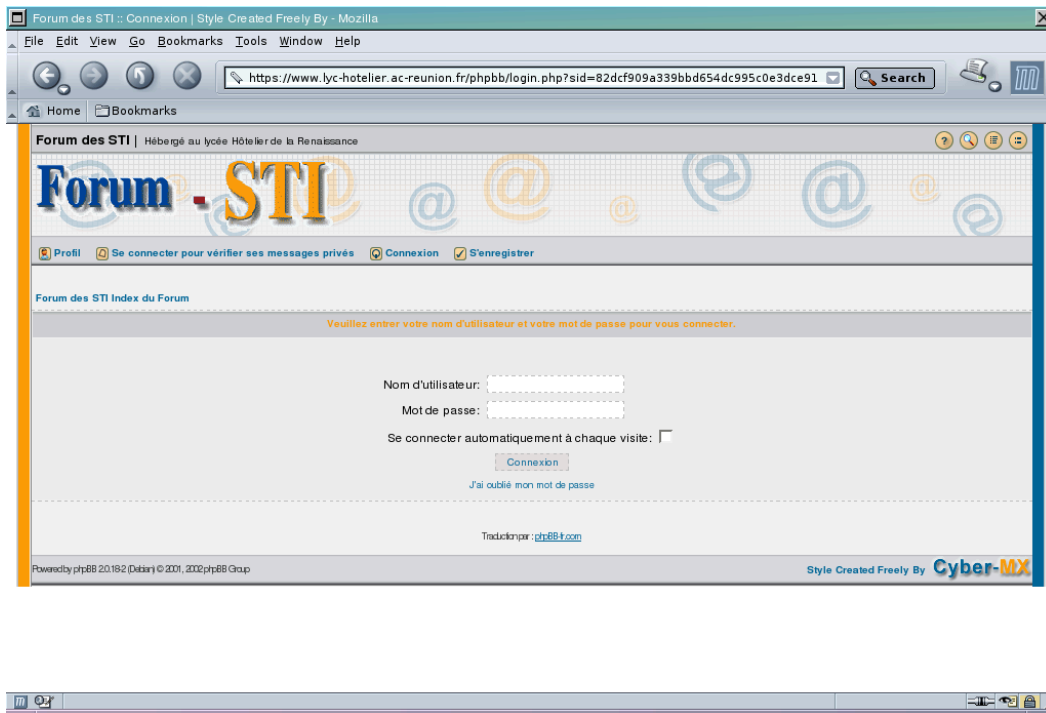
Il s'agit des fonctions dont vous vous servirez le plus avec les sessions. Voici tout de même, pour votre culture générale, quelques fonctions qui apparaîtront de temps en temps.

2.1.3.4.3. `session_name([opt nom])`

Elle vous permet de déterminer le nom de la variable transmise via cookie ou via URL (par défaut `PHPSESSID`). Si vous lui passez un argument, elle changera le nom par défaut, à utiliser avant d'initialiser une session.

Dans la capture d'écran suivante la fonction a été appelée comme suit :

```
1
2
3 <?php
4
5 /* choisi le nom de session : sid */
6 $previous_name = session_name("sid");
7
8 echo "L'ancien nom de la session était $previous_name<br />";
9
10 ?>
11
```

2.1.3.4.4. session_id([opt id])

Comme vous avez récupéré le nom de la variable, il peut être intéressant de connaître sa valeur non ? `session_id([opt id])` vous le permet. De même, si un argument lui est passé, elle redéfinit l'identifiant de session.

2.1.3.5. Exemples d'implémentation

2.1.3.5.1. Création d'un espace membre

Voilà une des principales applications des sessions, permettre à un utilisateur d'accéder à un espace différent après s'être identifié. Pour cet exemple, on considèrera que la variable `$is_identified` vaut 1 si le membre s'est correctement identifié via un formulaire.

```

1
2
3 session_start();
4
5 if (isset($_POST('is_identified')) {
6     $member = 1;
7     session_register('member');
8 }
9 else {
10  /* Affichage du formulaire d'identification */
11 }
12
13
```

Et pour l'accès aux fonctions de l'espace membre :

```

1
2
3 session_start();
4
```

```
5 if ($_SESSION['member'] == 1) {
6     /* Accès aux fonctions de l'espace membre */
7 }
8 else {
9     echo 'Il faut être identifié pour accéder à cette section';
10 }
11
12
```

2.1.3.5.2. Compteur de visiteurs online

Vous avez sûrement pu le voir, sur de nombreux sites, on peut voir le nombre de visiteurs en ligne sur le site. En fait, il ne s'agit pas de sessions très précises. Le protocole HTTP étant ce qu'il est, il est impossible de déterminer si un utilisateur est sur un site ou pas. Pour ce faire, on part du principe qu'un visiteur est encore en ligne X minutes après son dernier rafraîchissement de page.

Ainsi donc, nous allons compter le nombre de sessions *actives*. C'est à partir de là qu'il faut ruser ... En PHP, on ne peut pas déterminer le nombre de sessions via une fonction, mais il est possible de déterminer le nombre de fichiers de sessions. Il suffit juste de savoir si un fichier de session est encore valide selon nos critères, c'est à dire, si la date de dernière modification de session (rafraîchissement de page) est inférieure à un certain temps.

Sur toutes vos pages, il suffit de :

```
session_start();
```

La fonction du compteur :

```
1
2
3 if ( $d = opendir( session_save_path() ) ) {
4     $count = 0;
5     $session_timeout = 3 * 60;
6     while ( false !== ( $file = readdir( $d ) ) ) {
7         if ( $file != '.' && $file != '..' ) {
8             // plus lisible ...
9             if ( time() - filemtime(session_save_path() . '/' . $file) < $session_timeout ) {
10                 $count++;
11             }
12         }
13     }
14 }
15 echo $count;
16
17
```

Dans cet exemple, `$session_timeout` correspond à la durée en secondes pendant laquelle une session est considérée comme active. `$count` contient le nombre de sessions actives.

2.1.3.6. Sérialisation des données

C'est une fonction *magique* qui permet de transporter dans une chaîne de caractère une variable, un tableau ou un objet de script en script sans perdre ni leur structure, ni leur type.

```
string serialize( string value (string callback) );
```

Il est possible de définir une fonction de callback qui sera appelée si une classe indéfinie est utilisée lors de la délinéarisation (ce qui évitera de voir l'objet recevoir le type d'objet incomplet `object __PHP_Incomplete_Class`). Pour désactiver cette fonctionnalité, laissez la simplement vide.

Pour récupérer une variable linéarisée, et retrouver une variable, utilisez `unserialize()`. `serialize()` acceptent tous les types sauf les ressources. Vous pouvez linéariser un tableau qui contient une référence sur lui-même. Les références dans ce tableau/objet seront également stockées.

Exemple 2.1. Stockage des informations de sessions dans une base de données

```
1
2
3 // $session_data contient un tableau multi-dimensionnel , avec les
4 // informations de session de l'utilisateur courant. On utilise serialize()
5 // pour les stocker dans une base de données
6
7 $conn = odbc_connect ("webdb", "php", "chicken");
8 $stmt = odbc_prepare ($conn,
9     "UPDATE sessions SET data = ? WHERE id = ?");
10 $sqldata = array (serialize($session_data), $PHP_AUTH_USER);
11 if (!odbc_execute ($stmt, &$sqldata)) {
12     $stmt = odbc_prepare($conn,
13         "INSERT INTO sessions (id, data) VALUES(?, ?)");
14     if (!odbc_execute($stmt, &$sqldata)) {
15         /* Grosse bourde! Souffre et potasse! */
16     }
17 }
18
19
```

2.2. Évaluation

Voilà une batterie d'exercices, parfois la documentation que vous venez de parcourir sera suffisante, parfois une recherche sur *internet* sera nécessaire afin de préparer ce TP !

Bon courage !

2.2.1. TP1 Série PHP

Pour information voici un exemple de parcours d'un tableau associatif :

```
<?php
$nom["Thècle"]="Assicmonpote";
$nom["Édith"]="Avuleur";
$nom["Kelly"]="Diocy";
$nom["Mélusine"]="Enfaillite";
$nom["Elmer"]="Hitmieux";

reset($nom);

echo "<table>";
while (list($clef,$valeur)=each($nom)) {
    echo "<tr><td>$clef</td><td>$valeur</td></tr>\n";
}
echo "</table>";
?>
```

2.2.1.1. Exercice 1

Vous devez créer un lien escamotable. Ce lien apparait si *l'ordinateur naviguant* possède une adresse IP se situant dans une fourchette que vous aurez préalablement défini.

Si *l'ordinateur naviguant* n'a pas d'adresse IP contenue dans la fourchette prédéfinie le lien n'est plus visible, mais un accès direct reste possible.

Le nom du fichier, dans lequel sera appelée la fonction sera `lien.php`, le répertoire vers lequel pointe le lien est intranet. Vous déposerez un fichier `index.php` dans ce répertoire pour faire vos tests.

Afin de rendre ce script le plus facilement réutilisable vous créerez une fonction dont voici le prototype :

```
$reprotect="intranet";
$adrdeb="192.33.97.1";
$adrfin="192.33.97.15";

retractablelink ($reprotect,$adrdeb,$adrfin);
```

Les adresses IP de début et de fin devront correspondre à une adresse de classe C standard : les 3 premiers champs devront être identiques, et dans le cas contraire un message d'erreur devra apparaître indiquant au développeur qu'il s'est trompé dans l'appel de cette fonction. Il convient de vérifier également que l'adresse de début est bien inférieure à l'adresse de fin ...

2.2.1.2. Exercice 2

On dispose du fichier suivant `tableau-01.txt` contenant des adresses web (rien ne vous empêche d'en choisir d'autres) :

```
http://www.hoaxbuster.com/
http://www.gazel.nu/faqs/virus.htm#e-mail
http://www.electriccafe.org/JBT/
http://www.zetetique.ldh.org/
```

```
http://perso.wanadoo.fr/jean.brissonnet/  
http://www.thejackytouch.com/  
http://www.multimania.com/lepoulpe
```

Ecrire un script `filetolink_v1.php` qui lit ce fichier pour construire une page web contenant une liste de liens hypertextes.

2.2.1.3. Exercice 3

Même exercice, mais cette fois chaque ligne du fichier `tableau-02.txt` comprend aussi une description du site pointé. La séparation étant assurée par la chaîne `"**"` :

```
http://www.hoaxbuster.com/**HoaxBuster, première ressource francoph...  
http://www.gazel.nu/faqs/virus.htm#e-mail**Est-il possible d'attrap...  
http://www.electriccafe.org/JBT/**Nemo Joe Bar Team Spirit  
http://www.zetetique.ldh.org/**Le cercle Zététique  
http://perso.wanadoo.fr/jean.brissonnet/**Éthique et toc  
http://www.thejackytouch.com/**The Jacky Touch  
http://www.multimania.com/lepoulpe**Le poulpe sur la toile
```

Ecrire un script `filetolink_v2.php` qui lit ce fichier pour construire dans un tableau une liste de liens hypertextes ainsi que leurs descriptions.

2.2.1.4. Exercice 4

Même exercice, mais cette fois chaque description et l'adresse correspondante (fichier `tableau-03.txt`) sont sur deux lignes consécutives :

```
HoaxBuster, première ressource francophone sur les hoax  
http://www.hoaxbuster.com/  
Est-il possible d'attraper un virus en ouvrant un e-mail?  
http://www.gazel.nu/faqs/virus.htm#e-mail  
Nemo Joe Bar Team Spirit  
http://www.electriccafe.org/JBT/  
Le cercle Zététique  
http://www.zetetique.ldh.org/  
Éthique et toc  
http://perso.wanadoo.fr/jean.brissonnet/  
The Jacky Touch  
http://www.thejackytouch.com/  
Le poulpe sur la toile  
http://www.multimania.com/lepoulpe
```

Ecrire un script `filetolink_v3.php` qui lit ce fichier pour construire dans un tableau une liste de liens hypertextes ainsi que leurs descriptions.

2.2.1.5. Exercice 5

On donne une liste de personnes dans le fichier `tableau-04.txt` :

```
1;Thor;Aipaleur;tata  
2;Dick;Sionnaire;dsds
```

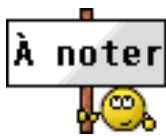
```
3;Debbie;Zoudanlkou;dzd
4;Mélanie;Zaitofrai;mzmz
5;Helmut;Ardelpic;haha
6;Jacques-André;Lejouré-Lanuit;jljl
7;Phil;Alapate;papa
```

Chaque ligne est composée, dans l'ordre, d'un identifiant (un nombre), un prénom, un nom, et un mot de passe. Écrire une unique page web `affiche_identifiant.php` qui donne à sélectionner (en utilisant un champ `<select>`) une des personnes et qui affiche ensuite son mot de passe.

2.2.1.6. Exercice 6

En vous basant sur le fichier `tableau-04.txt` développer une fenêtre d'authentification (fichier `auth_simple.php`) proposant dans une liste déroulante le nom et une entrée pour la saisie du mot de passe.

Dans la même page `auth_simple.php`, si le nom et le mot de passe saisis sont valides alors afficher un message indiquant que la connexion a réussi. Si, l'authentification échoue alors renvoyer sur le formulaire du départ.



Un switch/case peut-être ?

Ici, comme dans l'exemple précédent, il n'y a qu'un seul fichier php `auth_simple.php`, vous devez donc vérifier lors de son affichage dans quel cas vous vous trouvez.

2.2.1.7. Exercice 7

En vous basant sur l'exercice précédent, ajouter une authentification **md5**. Les mots de passes **md5** seront contenus dans un fichier `tableau-05.txt`.

```
1;Thor;Aipaleur;375a52cb87b22005816fe7a418ec6660
2;Dick;Sionnaire;fdcd0ec8cfe1e177670f658ae15721e
3;Debbie;Zoudanlkou;13085a63a2b3e4beb7ab10ee395aefe4
4;Mélanie;Zaitofrai;7694f4a66316e53c8cdd9d9954bd611d
5;Helmut;Ardelpic;21232f297a57a5a743894a0e4a801fc3
6;Jacques-André;Lejouré-Lanuit;0cc175b9c0f1b6a831c399e269772661
7;Phil;Alapate;papa;2a76230ba6c042409e5b010fcb5fc14b
```



Ce sont des exemples de md5 ()

Attention, ici les mots de passes cryptés sont donnés à titre d'exemple. Ils ne correspondent pas aux mots de passe de l'exercice précédent.

2.2.2. TP2 les Cookies

2.2.2.1. Exercice 1

L'objectif de cet exercice est de réaliser un compteur de visite basique à l'aide bien évidemment d'un cookie.

Lorsque l'utilisateur arrive sur la page `/cookies/exo1/index.php`, la page web affiche un petit message : "Bienvenue ! Vous avez effectué sur ce site : **n** visite(s)".

Vous devrez également gérer le pluriel de *visite*. En affichant : *visite* ou *visites*.



Il y a exercice et réalité !

Si vous **deviez** réaliser un compteur de visite digne de ce nom, il vous faudrait bien évidemment le gérer en fonction de l'adresse IP de la machine cliente. L'actualisation du compteur ne devant pas se faire sur un *réactualisé* page. Cette fonctionnalité n'est pas à implémenter ici.

2.2.2.2. Exercice 2

Ecrire un script `/cookies/exo2/index.php` qui vérifie la possibilité d'utiliser les cookies.

Cette vérification doit s'effectuer en deux phases :

1. Envoyer une requête set-cookie puis recharger la page active. Pour recharger automatiquement la page vous utiliserez la balise meta `http-equiv="refresh" content="0; ...` le zéro représente le délai en seconde avant la redirection. Voici un exemple de redirection :

```
<html>
<head>
  <meta http-equiv="refresh" content="0; url=nouvelleepage.php">
  <title>Redirection ...</title>
</head>
<body>
  ...
</body>
</html>
```

2. Vérifier si la requête set-cookie a réussi, effectuer une action en fonction du test précédent en affichant : *Les cookies sont pris en compte sur le client* ou *Les cookies ne sont pas pris en compte sur le client*.

2.2.2.3. Exercice 3

L'objectif de cet exercice `/cookies/exo3/index.php` est d'implémenter la possibilité d'utiliser un tableau de cookies pour y stocker plusieurs valeurs.

Pour cela, vous créerez un formulaire permettant de saisir les champs suivants : le nom Username, l'âge Age, le sexe Sexe et l'adresse E-MAIL. Vous proposerez une méthode pour la saisie correcte de l'adresse E-MAIL.

Lorsque l'internaute s'est correctement identifié *en remplissant tous les champs* vous poserez un tableau de cookies chez le client. Lors d'un prochain accès, ses informations personnelles devront s'afficher à la place du formulaire.

2.2.2.4. Exercice 4

Réaliser un formulaire à deux champs, l'un pour le login et l'autre pour le mot de passe. L'internaute renseigne les deux champs, s'ils correspondent à respectivement 'admin' pour le login et 'pass' pour le mot de passe cette même page affiche :

Vous êtes authentifié sous : admin

S'ils correspondent à respectivement 'guest' pour le login et 'coucou' pour le mot de passe cette même page affiche :

Vous êtes authentifié sous : guest

Dans les deux cas un lien apparait proposant le retour au formulaire et les deux champs doivent être alors préremplis des valeurs précédentes (login et mot de passe) si l'internaute y revient avant 10 secondes, sinon les deux champs sont initialisés à vide.

2.2.3. TP3 les Sessions

2.2.3.1. Exercice 1

L'objectif de cet exercice est de réaliser le changement dans la couleur de fond d'une page Web suivant l'utilisateur authentifié et pour une page bien définie.

Dans ce fichier `/session/exo1/auth.php` vous proposerez une fenêtre d'authentification. Login uniquement !

Si l'utilisateur est correctement authentifié le formulaire de saisie est remplacé par un lien vers la page `/session/exo1/un.php`. Sinon, le formulaire doit réapparaître proposant ainsi une nouvelle saisie.

Vous coderez *en dur* un profil *l'admin*, l'utilisateur ordinaire étant un internaute non logué.

Si l'utilisateur ordinaire accède directement à la page `/session/exo1/un.php` celle-ci devra avoir un fond bleu, si c'est *l'admin* le fond de page sera rouge.

2.2.3.2. Exercice 2

L'objectif de cet exercice est de réaliser le changement dans la couleur de fond d'une page Web suivant l'utilisateur authentifié et pour une page bien définie.

Dans ce fichier `/session/exo2/auth.php` vous proposerez une fenêtre d'authentification. Login uniquement !

Si l'utilisateur est correctement authentifié le formulaire de saisie est remplacé par un lien vers la page `/session/exo2/un.php`. Sinon, le formulaire doit réapparaître proposant ainsi une nouvelle saisie.

Vous coderez *en dur* deux profils *l'admin* et l'utilisateur invité *guest*, l'utilisateur ordinaire étant un internaute non logué.

Si l'utilisateur ordinaire demande directement la page `/session/exo2/un.php` celle-ci devra avoir un fond bleu, si c'est *l'admin* le fond de page sera rouge, enfin si c'est *guest* le fond de page sera vert.

2.2.3.3. Exercice 3

L'objectif de cet exercice est de réaliser le changement dans la couleur de fond d'une page Web suivant l'utilisateur authentifié et pour une page bien définie. De plus, si l'utilisateur n'est pas authentifié, il ne pourra pas accéder aux pages ...

Dans ce fichier `/session/exo3/auth.php` vous proposerez une fenêtre d'authentification. Login uniquement !

Si l'utilisateur est correctement authentifié le formulaire de saisie est remplacé par un lien vers la page `/session/exo3/un.php`. Sinon, le formulaire doit réapparaître proposant ainsi une nouvelle saisie.

Vous coderez *en dur* deux profils *l'admin*, l'utilisateur invité *guest*, l'utilisateur ordinaire étant un internaute non logué.

Si l'utilisateur ordinaire demande directement la page `/session/exo3/un.php` celui-ci sera redirigé sur la page `/session/exo3/auth.php`, si c'est *l'admin* le fond de page sera rouge, enfin si c'est *guest* le fond de page sera vert. Donc, la page `/session/exo3/un.php` ne sera accessible que si l'utilisateur est authentifié.

2.2.3.4. Exercice 4

L'objectif de cet exercice est de réaliser une authentification basée sur une session.

Vous créerez un fichier texte (`/session/exo4/txt/auth.txt`) contenant des logins et mots de passes cryptés md5(), ainsi que les fichiers `/session/exo4/un.php`, `/session/exo4/deux.php`, `/session/exo4/trois.php` et `/session/exo4/auth.php`.

admin Il a le droit d'accéder à toutes les pages.

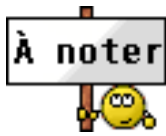
guest Peut accéder à /session/exo4/un.php.

middle Peut accéder aux pages /session/exo4/un.php et /session/exo4/deux.php.

Exemple 2.2. /session/exo4/txt/auth.txt

```
admin,4c9310d1375f9dbfdc6d2dffa2ea9dd07,106a6c241b8797f52e1e77317b96a201
guest,375a52cb87b22005816fe7a418ec6660,fdcde0ec8cfe1e177670f658ae15721e
middle,f15779c65bf7141196d01ae83f19ac83,2a76230ba6c042409e5b010fcb5fc14b
```

Si l'authentification n'a pas été faite les pages /session/exo4/un.php, /session/exo4/deux.php et /session/exo4/trois.php doivent renvoyer l'internaute automatiquement sur la page /session/exo4/auth.php. Cette page contient un formulaire avec une liste déroulante permettant de choisir le profil et deux zones de saisie permettant de taper respectivement un login et un mot de passe.



Ce sont des exemples de md5 () !

Penser à tester correctement l'authentification suivant le profil ! Le fichier texte /session/exo4/txt/auth.txt est donné ici à titre d'exemple, ce sera à vous de déterminer les *cryptés*.

2.2.4. TP4 synthèse

2.2.4.1. Généralités

Pour les exercices de synthèses suivants, vous devez rédiger un compte rendu dans lequel vous détaillerez :

- Votre démarche d'analyse.
- Les solutions choisies.
- Les algorithmes.
- Les codes sources commentés.
- Les modifications éventuelles.
- Les résultats obtenus.
- Les améliorations possibles.
- Vos conclusions.



Situation

Vous devez permettre l'accès à vos pages web avec l'url suivant : http://192.33.97.199/eleves/VOTRELOGIN/REPERTOIRE_DE_TRAVAIL. Vous disposez également d'un accès FTP passif avec vos login/password habituels. Pour ce qui est de la base de données, vous y avez accès via l'application phpmyadmin (<http://192.33.97.199/phpmyadmin/>) avec la même paires login/password dont vous disposez pour l'accès FTP.

2.2.4.2. Exercice 1 : les statistiques d'un site

2.2.4.2.1. Sujet

Nous avons vu que les sessions peuvent servir à protéger une partie d'un site. Ici, je vous propose une autre utilisation : faire des statistiques. Cela permet entre autres de connaître quelle page a vu le visiteur, combien de temps il est resté sur chaque page, savoir sur quel site il a cliqué pour venir sur le vôtre, etc ...

Vous devrez travailler dans le répertoire suivant : `/synthese/stats`.

Indiquer la méthode que vous devez utiliser ici afin de lancer automatiquement la session lors de l'appel d'une page par un internaute.

Les informations sur le visiteur vont être stockées dans une table de votre base de données. Celle-ci contiendra les champs suivants :

id	L'auto incrément.
sessid	Contient le numéro de session unique généré par php.
referer	Il s'agit de l'url du site sur lequel le visiteur a cliqué pour venir sur votre site.
parcours	Contient les url des pages demandées. Les url seront séparées par des points-virgules.
time	Contient les heures de chargement de chaque page (elles aussi séparées par des points-virgules).
navig	Le nom du navigateur utilisé par le client web.
os	Le système d'exploitation du poste client.
ip	L'adresse IP du poste client.

Il se peut que vous soyez amené à rajouter des champs. Merci de bien vouloir le justifier dans le compte rendu.

Pour les deux premières informations, ce sont les variables d'environnement qui vont vous servir. Il s'agit d'informations envoyées par le navigateur lorsque vous vous rendez sur un site. Php met entre autres à disposition : `$_SERVER['HTTP_REFERER']` et `$_SERVER['REQUEST_URI']`. Ces variables sont accessibles à n'importe quel moment dans votre script php.

Indiquer quelle est la fonction qui vous permettrait de modifier le nom de la session. Merci de préciser dans le compte rendu les raisons qui vous ont amené à choisir ou non cette fonction. Si vous pensez l'utiliser vous devez indiquer la raison dans le compte rendu.

Pour l'heure à laquelle la page a été demandée, vous allez la récupérer à l'aide de la fonction `time()`. Ceci vous permet à la fois de connaître le jour de la visite, ainsi qu'entre chaque page, en faisant une simple différence, de connaître en secondes combien de temps le visiteur est resté sur une page.

2.2.4.2.2. Etude du problème

Vous proposerez une étude complète du problème en joignant au compte rendu tous les algorithmes associés.

2.2.4.2.3. Méthode de test

Vous indiquerez en justifiant vos choix le nombre de pages que vous allez créer afin que vous puissiez tester correctement votre produit. Il vous est également demandé de définir les tests que vous mettrez en place.

2.2.4.2.4. Affichage des statistiques

Vous proposerez et implémenterez une fonction permettant d'afficher les statistiques dans un tableau. L'adresse IP du navigateur client sera un lien pointant vers une page de consultation indiquant le nombre de pages vu par le client, ainsi que temps passé par page.

2.2.4.3. Exercice 2 : un logiciel de chat

2.2.4.3.1. Sujet

Ici, vous devez réaliser une application PHP permettant d'offrir aux internautes un **chat** basique. Les messages seront enregistrés dans une base de données.

Veuillez indiquer en vous basant sur le cours la nécessité ou non d'implémenter ici les sessions.

On souhaite avoir, sur la même page, deux zones de texte. L'une en haut pour écrire votre pseudo, l'autre en dessous pour écrire votre petit message. Ensuite, un bouton **Envoyer** permettra d'envoyer les données à MySQL, pour qu'il les enregistre dans une table de la base de données.

En-dessous, le script devra afficher les 10 derniers messages qui ont été enregistrés (si vous les affichez tous et que vous avez 1000 messages ça risque d'être un peu long !) en allant du plus récent au plus ancien.

2.2.4.3.2. Etude du problème

Vous proposerez une étude complète du problème en joignant au compte rendu tous les algorithmes associés.

2.2.4.3.3. Améliorations

Vous devez également, après tests, proposer au moins 3 *pistes* pour améliorer le produit **chat**.

2.2.5. TP5 : Gestion d'un caddie en PHP

2.2.5.1. Objectifs

Réalisation pas à pas d'un "panier d'achats", brique indispensable de tout site de commerce électronique. Vous devrez coder en PHP le fonctionnement de ce caddie électronique ...

Caddie, panier, chariot... Les termes sont nombreux pour désigner une seule et même chose : l'emplacement où le site Web stocke une sélection d'articles, généralement en prévision d'un "passage à la caisse".

Un caddie sert deux types de visiteurs : ceux qui sont enregistrés sur le site, et ceux qui ne le sont pas (encore). Dans les deux cas, le fonctionnement du caddie est le même : l'utilisateur parcourt le catalogue disponible sur le Web, choisit les produits qu'il désire acheter, et en fin de parcours, regarde une dernière fois le contenu de son caddie afin d'y apporter quelques éventuelles modifications avant de valider l'achat et de payer le tout.

2.2.5.2. L'art des choix

Notre souci premier sera de conserver les données envoyées par le visiteur pendant sa présence sur le site (ou "session client") : en effet, il ne fait que passer de page en page, sélectionnant ici ou là un ou plusieurs produits - il s'attend logiquement à tous les retrouver dans son panier. Pour cela, nous avons deux solutions : les cookies, ou les sessions serveur.

Les cookies sont de petits fichiers textes, stockés par le serveur sur l'ordinateur du client. Un cookie peut contenir jusqu'à 4 Ko d'informations, et un serveur ne peut normalement envoyer que 20 cookies par client. Par ailleurs, un ordinateur n'est pas sensé accepter plus de 300 cookies. Sa durée de vie peut varier de la durée de la session client (jusqu'à la fermeture du navigateur) à plusieurs années...

4 Ko permet de stocker un bon nombre d'informations, mais il peut toujours arriver une commande dépassant cette taille. Pire : les clients peuvent parfaitement choisir de ne pas accepter certains cookies ...

Les sessions sont probablement le meilleur choix. Lorsqu'un site utilise des sessions, chaque visiteur reçoit un identifiant unique, qui correspond au nom du fichier, stocké sur le serveur, contenant les informations enregistrées par celui-ci. La taille du fichier est (en théorie) illimitée, il le suit durant la totalité de la session, et même après : la durée de vie d'une session est généralement de 24 minutes, donc si l'utilisateur ferme le navigateur par inadvertance, il pourra retrouver sa session en se reconnectant...

2.2.5.3. Utilisation des sessions

Les informations relatives aux articles sélectionnés sont donc stockées dans une session serveur, ou plus précisément dans des variables de session.

Utiliser une session n'est pas très compliqué : il suffit d'ajouter au début de chaque page les utilisant (donc, le catalogue) le code suivant :

```
1
2
3 <?php session_start(); ?>
4
5
```

Cela initialise la session (ou retrouve une session sur le serveur).

Notre session, tout du long du cheminement du (futur) client, contiendra une variable dédiée uniquement à la commande en train de se construire : un tableau associatif, \$achats[x][y]. Le tableau "x" ne contiendra qu'une valeur, le numéro du produit dans la commande, tandis que le tableau "y" contiendra l'id du produit, et la quantité voulue. Par exemple, écrit directement en PHP, nous aurions pour un seul produit :

```
1
2
3 <?php
4     $achats = array (
```

```

5      "1" => array("id" => 135423, "qte" => 2)
6    );
7  ?>
8
9

```

En travaillant avec les sessions, cela prendrait la forme suivante :

```

1
2
3 <?php
4  session_start();
5  $_SESSION['achats'] = array ("id" => 135423, "qte" => 2);
6  ?>
7
8

```

L'utilisation de ces variables de session se fait ensuite via la variable \$_SESSION :

```

1
2
3 <?php
4  $id = $_SESSION['achats']['id'];
5  $qte = $_SESSION['achats']['qte'];
6  ?>
7
8

```

2.2.5.3.1. Définition des actions

Lors de l'utilisation d'un caddie, l'utilisateur à quelques commandes de base : **ajouter un produit dans le caddie** (utilisé pendant le parcours du catalogue), **retirer un produit de la commande** (utilisé lors de l'affichage final), **actualiser le caddie** (idem, affichage final - pour les clients qui voudraient ajouter ou enlever des produits) et enfin, **valider la commande**. Chacune de ses commandes fera l'objet d'un script au sein de notre application.

2.2.5.3.1.1. Ajouter

Nous partons du principe que nous sommes dans le catalogue, et que plusieurs produits sont affichés. L'affichage des produits est bien entendu dynamique (selon la recherche ou la rubrique).

Figure 2.1. Le fichier catalogue.php

Votre Boutique en ligne					
Nom	Prix	Description	Photo	Qte	Ajout
Porte encens "Paon"	10	Porte encens de temple en forme de Paon avec plateau pour récupérer les cendres. Il est possible avec ce porte encens de faire brûler trois bâtons à la fois. Le Paon est symbole de sagesse et de force, 100% laiton Dimensions : 25cm		3 ▼	Ajouter
Porte encens bois	3	Porte encens en bois avec plateau pour récupérer les cendres, 100% bois de belle qualité avec design épuré. Dimensions : 27cm de longueur x 2.5cm		1 ▼	Ajouter
Brûle résine d'encens Cuivre	28	Brûle résine d'encens de très belle qualité avec une grille pour le charbon. En cuivre. Hauteur : 13 cm - Diamètre : 6,5 cm		1 ▼	Ajouter

On veut que toute l'interaction se fasse à partir du lien situé sous le texte *Ajouter* quand on le clique. On veut de plus que les références du produit de ce champ (son id et la quantité souhaitée) soient ajoutées à notre variable \$liste[[]], et que le catalogue s'affiche à nouveau, à la même page, avec la valeur des quantités mises à jour. Ce lien renvoie donc sur la même page catalogue.php.

Le code PHP gérant l'ajout prendra cette forme :

```

1
2
3 <?php
4 if ($_GET['ajout'] == "AJOUT")
5 {
6     $_SESSION['achats'][] = array (
7         "id" => $_GET['id'],
8         "qte" => $_GET['qte']
9     );
10 }
11 ?>
12
13

```

Le lien Ajouter correspondra donc à : `catalogue.php?ajout="AJOUT"&id=123456&qte=3`

2.2.5.3.1.2. Supprimer

Nous travaillons maintenant dans notre caddie (`caddie.php`). Il se présente ainsi:

Figure 2.2. Le fichier `caddie.php`

Produit	Prix	Quantité	Total		
PremierProduit	127	3	381	Mettre à jour	Retirer
UnBonProduit	450	5	2250	Mettre à jour	Retirer
Prix total		8 produit(s)	2631	Valider la commande	

Nous laissons les calculs et affichages des totaux, qui ne font qu'utiliser les même méthodes que pour l'affichage du catalogue, l'arithmétique en plus. L'affichage des quantités dans le menu déroulant est intelligemment géré par l'utilisation de *selected*.

Le code PHP gérant le menu déroulant par l'utilisation de *selected* :

```

1
2
3 <?php
4 for ($i=0; $i < count($_SESSION['achats']); $i++)
5 {
6     // code MySQL allant chercher les informations pour chaque produit
7     // en fonction de $_SESSION['achats'][$i]['id']
8     ?>
9     <select name="qte<?php $i ?>">
10         <option value="1"
11             <?php ($_SESSION['achats'][$i]['qte'] == 1 ? "SELECTED" : null;?>
12         </option>
13         <option value="12"
14             <?php ($_SESSION['achats'][$i]['qte'] == 2 ? "SELECTED" : null;?>
15     <!-- etc. -->
16     </option>
17 </select>
18 <?php }
19
20 ?>
21
22

```

Nous travaillons toujours avec nos variables de session : pour supprimer un élément de `$_SESSION['achats']`, il nous faut la parcourir, à la recherche de l'id du produit à supprimer. Notre lien *Mettre A Jour* sera de la forme suivante : `caddie.php?retirer=RETIRER&id_produit=123456`

Nous ferons cela à l'aide d'une boucle `for` et de la fonction `array_splice()`, qui permet d'effacer une portion d'un tableau :


```

1
2
3 <?php
4 if ($_GET['retirer'] == "RETIRER")
5 {
6     for ($i=0; $i < count($_SESSION['achats']); $i++)
7     {
8         if ($i == $_GET['id_produit'] )
9         {
10             array_splice($_SESSION['achats'], $i, 1);
11         }
12     }
13 }
14 ?>
15
16

```

2.2.5.3.1.3. Mettre à jour

Sur le même modèle que Supprimer : on charge la même page, en mettant simplement à jour les valeurs de session...

2.2.5.3.1.4. Valider

Cette dernière commande renvoi la page d'affichage final, où l'utilisateur ne peut plus modifier sa liste d'achats, ni revenir en arrière. Les valeurs sont sauveées dans la base MySQL, sur le schéma suivant : il est proposé à l'utilisateur de se loguer afin de valider ses achats, sinon il peut créer un compte lui permettant de plus tard de valider ses achats.

En exemple ici la partie de code qui permet d'enregistrer les achats en base de données :

```

1
2
3 <?php
4 $req = "INSERT INTO commandes (id_client, date, total, etat)
5     VALUES (xborderie, " date(); . " , " . $_GET['total'] . " , "En cours...");"
6
7 mysql_query($req);
8 $commande_id = mysql_insert_id();
9
10 for ($i=0; $i < count($_SESSION['achats']); $i++)
11 {
12     $req2 = "INSERT INTO achats (id_produit,id_commande,quantite,prix_total)
13     VALUES ( " . $_SESSION['achats'][$i]['id'] . " , " . $commande_id . " . " , "
14     $_SESSION['achats'][$i]['qte'] . " , " . $_SESSION['achats'][$i]['qte'] * $_SESSION['achats']
15     [$i]['prix'] . " );";
16     mysql_query($req2);
17 }
18 ?>
19
20

```

Chapitre 3. Évaluation TD & TP du module M2105

Table des matières

3.1. Évaluation	54
3.1.1. TD1 : Analyse d'un système de gestion de commandes	55
3.1.1.1. Etude du système de gestion de commandes (MCD)	55
3.1.1.2. Réalisation de l'application PHP	56
3.1.2. TD2 : Incorporer un framework CSS & javascript au système de gestion de commandes	59
3.1.2.1. Installation de Foundation	59
3.1.2.2. Configuration de Foundation	60
3.1.3. TD3 : Amélioration et finalisation du système de gestion de commandes	64
3.1.3.1. Amélioration du code PHP	64
3.1.3.2. Utilisation d'Ajax	65
3.1.3.3. Ce qui reste à faire !	70
3.1.4. TP1 : <i>Le Grand Hôtel</i> - MCD et requêtes SQL	72
3.1.4.1. Modifier le modèle MCD (AnalyseSI)	73
3.1.4.2. Définir des requêtes SQL	74
3.1.5. TP2 : Foundation intégration	76
3.1.5.1. Quelques points techniques	78
3.1.6. TP3 : Gestion des comptes utilisateur	81
3.1.6.1. Quelques points techniques	82
3.1.7. TP4 : La réservation	85
3.1.7.1. L'algorithme permettant de renseigner la table 'Lignefacture'	86
3.1.7.1.1. Votre travail	88
3.1.8. TP5 : Choix de la chambre et facturation	90
3.1.8.1. Choix de la chambre	90
3.1.8.2. La facturation	91
3.1.9. TP6 : Synthèse	92
3.1.9.1. La mise au point de la BDD	92
3.1.9.2. La partie administration du site	92
Références	92

3.1. Évaluation

Voilà une batterie d'exercices, parfois la documentation que vous venez de parcourir sera suffisante, parfois une recherche sur *internet* sera nécessaire afin de préparer ce TP !

Bon courage !

3.1.1. TD1 : Analyse d'un système de gestion de commandes

A l'aide du logiciel AnalyseSI vous réaliserez l'étude d'un système de gestion de commandes pour un magasin. Bien entendu, vous prendrez *InnoDB* comme moteur de la BDD MySQL. Puis vous le coderez en PHP.

Objectifs :

- Réaliser le modèle MCD à l'aide du logiciel AnalyseSI ;
- Importer le modèle généré dans MySQL ;
- Entrer des valeurs fictives afin d'alimenter la base ;
- Définir les contraintes sur les clés étrangères entre les tables et les tester ;
- Vous créez un *dump* de la BDD.
- Enfin, en vous aidant du code *pdo-master* vous réaliserez la création automatique de la BDD, puis les formulaires : d'ajout de client, de produit et de commande.

Déroulement.

- Travailler par groupe de 2 étudiants (maxi 3 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 2 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TD, la suite (examen final + note de TP) dépend de votre compréhension globale.



Les autres TD ... Et les TP ...

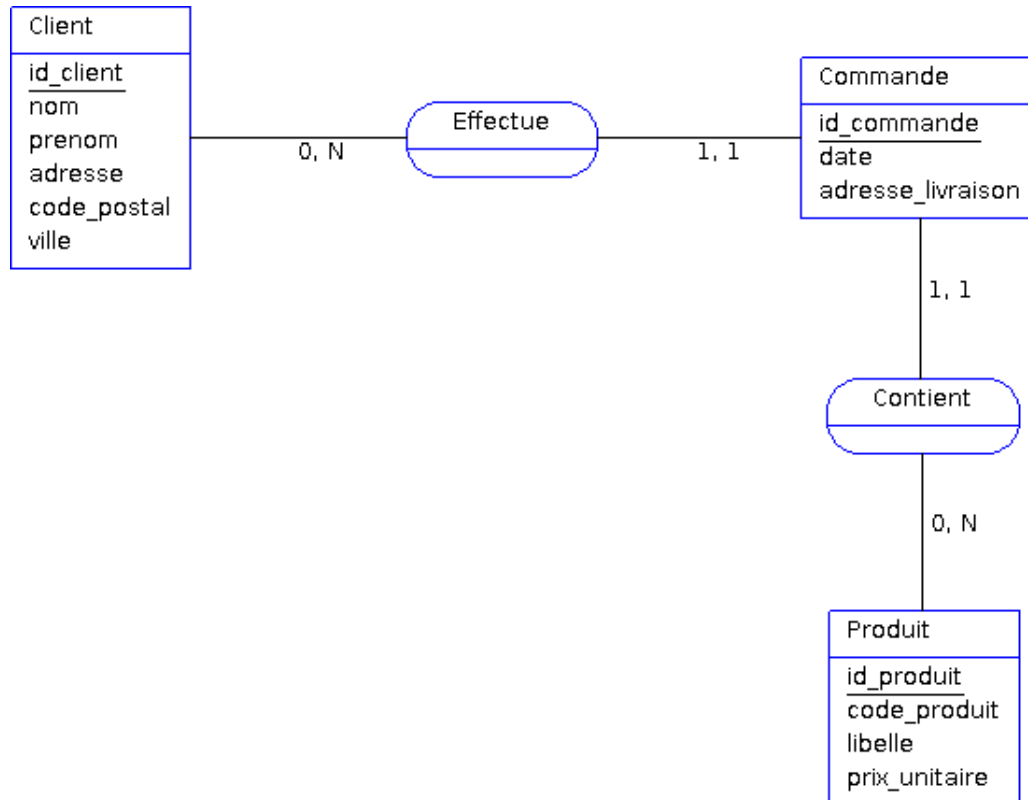
Les deux autres TD dépendent du TD1, ils sont la suite ... Les TP c'est différent, c'est pas le même problème qui est posé, mais tout le code que vous aurez fourni en TD vous sera utile.

3.1.1.1. Etude du système de gestion de commandes (MCD)

A la fin de cette étude vous devez fournir un *dump* de la BDD (en utilisant phpmyadmin) qui sera l'entrée de la réalisation qui suit.

Les règles de gestion :

- Le magasin vend des produits à des clients ;
- Les produits possèdent une référence (un code), un libellé et un prix unitaire ;
- Les clients ont une identité (nom, prénom, adresse ...) ;
- Les clients passent des commandes de produits. On mémorise la date de la commande (0 à N commandes) ;
- Pour chaque commande, le client précise une adresse de livraison ;
- La commande concerne un produit ;
- Un produit est concerné par 0 à N commandes.



Les contraintes sur les clés étrangères :

- Si je supprime un client (de la base) la commande qu'il a effectuée est aussi supprimée ;
- Si je mets à jour le champ id_client de la table client, le champ correspondant de la table commande est mis à jour automatiquement ;
- Si je supprime un produit (de la base) la commande qu'il a effectuée est aussi supprimée.
- Si je mets à jour le champ id_produit de la table produit, le champ correspondant de la table commande est mis à jour automatiquement ;

3.1.1.2. Réalisation de l'application PHP

Je vous conseille vivement de suivre la démarche proposée ci-dessous. Vous pouvez récupérer le code de *pdo-master* ici : GITHUB [<https://github.com/taniascia/pdo/tree/feba19a43e63c617c95a1f6e68825f6e3086336c>] ou sur Moodle.

Démarche à suivre :

- Vous avez un script dans la machine virtuelle permettant de créer un compte de base de données, utilisez le !
- Avec phpmyadmin créer une table fictive (avec des champs ressemblant à ce qui est demandé ...) faites en un *dump* (ceci permet de travailler sans avoir encore la première partie du TD de fonctionnelle) ;
- Tester la création automatique de la BDD avec le code proposé dans *pdo-master* ;
- Pour votre compréhension lisez les explications données sur *pdo-master* sur le site [<https://www.taniascia.com/create-a-simple-database-app-connecting-to-mysql-with-php/>] ;
- Réaliser les formulaires d'ajout de client, de produit et de commande en vous basant sur le code existant. Ci-dessous, les copies d'écrans de la réalisation des différents formulaires.

Figure 3.1. Formulaire principal



Figure 3.2. Formulaire d'ajout d'un client

The screenshot shows the 'Ajout d'un client' form. The heading is 'Système simple de gestion d'un magasin' followed by 'Ajout d'un client'. The form contains the following fields:

Nom

Prenom

Adresse

Code postal

Ville

[Retour](#)

Figure 3.3. Formulaire d'ajout d'un produit

The screenshot shows a web browser window with the address bar displaying '127.0.0.1/magasin/public/ajouter_produ'. The page title is 'Système simple de gestion d'un magasin'. The main heading is 'Ajouter un produit'. Below this, there are three input fields: 'Code produit', 'Libelle', and 'Prix unitaire'. A 'Submit' button is located to the right of the 'Prix unitaire' field. A purple link labeled 'Retour' is positioned below the 'Submit' button.

Figure 3.4. Formulaire permettant de commander

The screenshot shows a web browser window with the address bar displaying '127.0.0.1/magasin/public/commander.p'. The page title is 'Système simple de gestion d'un magasin'. The main heading is 'Commander un produit'. Below this, there are three input fields: 'Adresse de livraison', 'Nom du client', and 'Produit'. The 'Nom du client' field has a dropdown menu with 'Tudor' selected. The 'Produit' field has a dropdown menu with 'Feutre' selected. A 'Submit' button is located to the right of the 'Produit' field. A purple link labeled 'Retour' is positioned below the 'Submit' button.

3.1.2. TD2 : Incorporer un framework CSS & javascript au système de gestion de commandes

Maintenant que vous avez modifié *pdo-master* vous allez améliorer le rendu de celui-ci.

Objectifs :

- Installer Foundation (ZURB Foundation [https://foundation.zurb.com/]);
- Copier l'ancien *magasin* dans le *nouveau* ;
- Coder toutes les interfaces et formulaires en utilisant le fichier `index.html` ;
- Modifier les fichiers `header.php` et `footer.php` afin qu'ils tiennent compte de Foundation ;
- Modifier alors toutes les interfaces.

Déroulement.

- Travailler par groupe de 2 étudiants (maxi 3 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 2 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TD, la suite (examen final + note de TP) dépend de votre compréhension globale.



Les autres TD ... Et les TP ...

Les deux autres TD dépendent du TD1, ils sont la suite ... Les TP c'est différent, c'est pas le même problème qui est posé, mais tout le code que vous aurez fourni en TD vous sera utile.

3.1.2.1. Installation de Foundation

Installer Foundation n'est pas forcément une chose aisée, car il y a un grand nombre de méthode d'installation. Nous nous choisisons d'installer le script **foundation** permettant d'utiliser facilement par la suite le compilateur **sass**.

Quelques indications pour l'installation :

- Vérifier que la commande **foundation new** est installé, si oui vous passez directement à la suite ;
- Installer **npm** puis **foundation** :

```
[root@machina]# apt install npm
[root@machina]# npm i browser-sync --save
[root@machina]# npm rebuild node-sass
[root@machina]# npm install -g bower # pour pouvoir utiliser après : foundation new newqqc
[root@machina]# npm install --global foundation-cli
```



foundation n'est pas installé via apt

Mais via **npm** ...

3.1.2.2. Configuration de Foundation

Pour bien comprendre comment fonctionne Foundation, je vous conseille de regarder la vidéo sur le système de grille (C.F. XY Grid [<https://foundation.zurb.com/sites/docs/xy-grid.html>]). Ici vous devez utiliser le dernier système de grille disponible (inutile d'aller chercher à utiliser un autre système qui finira par devenir obsolète). Le système de grille est sûrement la chose la plus importante à comprendre afin d'être à l'aise avec les copiés/collés que vous finirez par faire inévitablement.

Démarche à suivre :

- Maintenant, je vous conseille de modifier le nom du répertoire de travail *de votre magasin* en `magasin_v0` au même niveau grâce à la commande **foundation new** vous allez créer un autre répertoire `magasin_v1`, celui-ci contiendra tous les fichiers (**foundation** + `magasin_v0`). Vous devez avoir cette arborescence :

```
[user@machina]$ pwd
/home/eleve/public_html/magasin      # ceci est un exemple !
[user@machina]$ ls
magasin_v0
[user@machina]$ foundation new

foundation new
? What are you building today? A website (Foundation for Sites)
? What's the project called? (no spaces) magasin_v1
? Which template would you like to use? Basic Template: includes a Sass compiler
```

Thanks for using ZURB Foundation for Sites!

Let's set up a new project.
It shouldn't take more than a minute.

Downloading the project template...

Done downloading!

Installing dependencies...

You're all set!

```
. New project folder created.
. Node modules installed.
. Bower components installed.
```

Now run foundation watch while inside the magasin_v1 folder.

```
[user@machina]$ ls
magasin_v0
magasin_v1
[user@machina]$ cd magasin_v1
```

- Comparez les noms des fichiers et répertoires puis renommez éventuellement certains fichiers afin que la copie se passe bien ;
- Par exemple faites ceci :

```
[user@machinal]$ pwd  
/home/eleve/public_html/magasin/magasin_v1  
[user@machinal]$ cp ../magasin_v0/* .
```

- Il est préférable d'utiliser la page `index.html` de Foundation afin de réaliser toutes les interfaces, formulaires et page d'accueil. Car en l'utilisant le navigateur est mise à jour automatiquement.
- Modifier dans un premier temps les fichiers `header.php` et `footer.php` afin qu'ils tiennent compte de Foundation. Pour cela, prenez exemple en regardant le fichier `index.html` ;
- Modifier les formulaires d'ajout de client, de produit et de commande en vous basant sur le code existant. Ci-dessous, les copies d'écrans de la réalisation des différents formulaires.

Figure 3.5. Formulaire principal

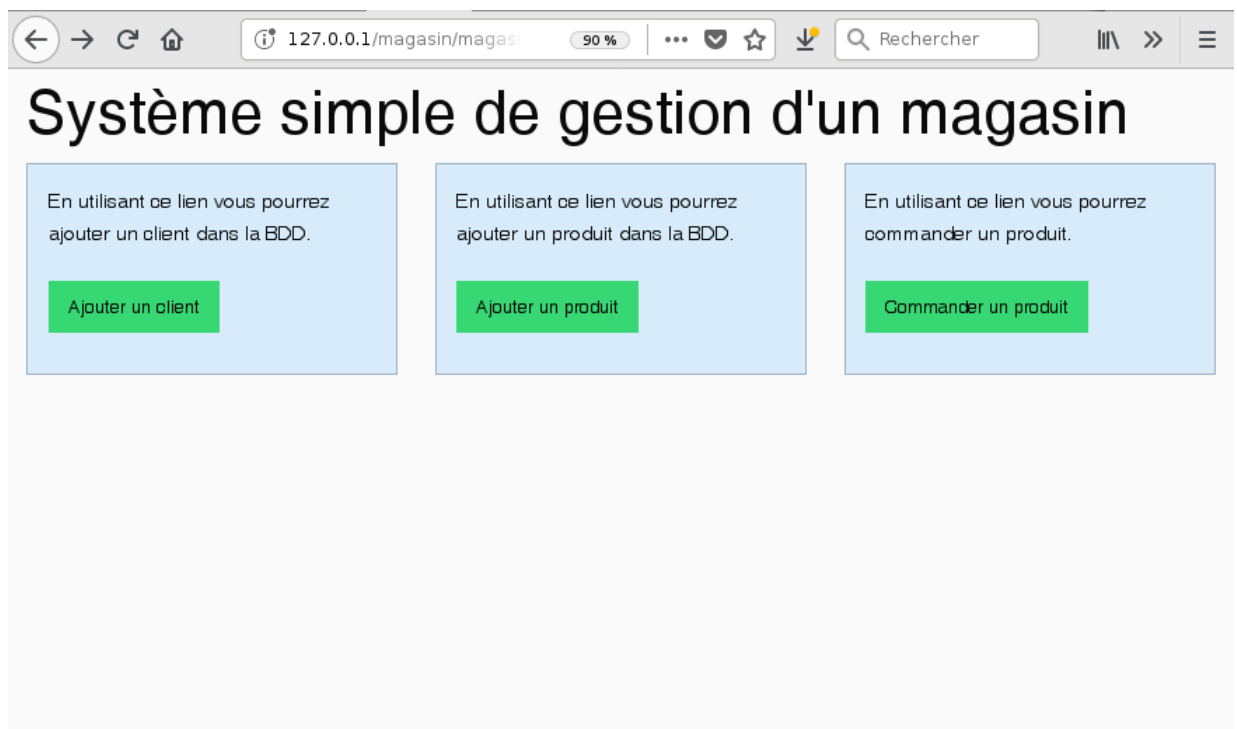


Figure 3.6. Formulaire d'ajout d'un client

The screenshot shows a web browser window with the address bar displaying '127.0.0.1/magasin/magasi' and a 90% zoom level. The page title is 'Système simple de gestion d'un magasin'. Below the title, the section 'Ajout d'un client' is visible. The form contains the following fields and values:

- Nom: MAGRON
- Prénom: Emanuel
- Adresse: Au champ ...
- Code postal: 75000
- Ville: PARIS

At the bottom right of the form is a green 'Valider' button. At the bottom left is a blue 'Retour' button.

Figure 3.7. Formulaire d'ajout d'un produit

The screenshot shows a web browser window with the address bar displaying '127.0.0.1/magasin/magasi'. The page title is 'Système simple de gestion d'un magasin'. The main heading is 'Ajouter un produit'. Below this, there are three text input fields: 'Code produit', 'Libelle', and 'Prix unitaire'. Each field contains a placeholder text with the same label. To the right of these fields is a green button labeled 'Valider'. To the left, there is a blue button labeled 'Retour'.

Figure 3.8. Formulaire permettant de commander

The screenshot shows a web browser window with the address bar displaying '127.0.0.1/magasin/magasi'. The page title is 'Système simple de gestion d'un magasin'. The main heading is 'Commander un produit'. Below this, there are three form elements: a dropdown menu for 'Produit' with 'Feutre' selected, a dropdown menu for 'Nom du client' with 'Tudor' selected, and a text area for 'Adresse de livraison' containing '120 route des fleurs ...' and 'CP : 97456 ...'. To the right of these elements is a green button labeled 'Valider'. To the left, there is a blue button labeled 'Retour'.

3.1.3. TD3 : Amélioration et finalisation du système de gestion de commandes

Oui, mais avons nous tout codé ? Avons nous terminé ? Est ce que tout cela est d'une qualité suffisante pour une utilisation professionnelle ? La réponse est non ! Dans ce cas, commencez à créer une copie de votre travail `magasin_v1 ...`

Objectifs :

- Dans un premier temps, modifier le code PHP, car il y a des choses qui font très mal aux yeux ... Des balises ouvrantes, suivies de balises fermantes, puis à suivre une autre ouverture de balise. Il faut limiter ce genre de pratique (C.F. document de cours) ;
- Après avoir nettoyé le code, il nous manque quelque chose : une interface de suppression des clients, produits ...
- Retrouver l'adresse de livraison d'une commande, il manque cette interface, en choisissant le client, une liste déroulante montrant tous les produits commandés par ce client est mise à jour automatiquement. En validant le formulaire, l'adresse de livraison s'affiche. Cela semble simple, mais la seconde liste déroulante est liée à la première, dans ce cas il faut utiliser une requête Ajax !
- On a pas du tout modifié le thème par défaut de Foundation et franchement c'est pas terrible !
- Utilisation d'un moteur de template comme Smarty (<https://www.smarty.net/docsv2/fr/> [https://www.smarty.net/docsv2/fr/]. **C'est pas une obligation**, mais si vous voulez tenter l'aventure, je valide à 100% ! J'en tiendrais compte inévitablement d'une manière ou d'une autre dans la note finale.
- Et après tout ça, vous aurez une bonne base pour aborder le feuilleton ... Je veux dire le TP en plusieurs étapes ...

Déroulement.

- Travailler par groupe de 2 étudiants (maxi 3 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 2 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TD, la suite (examen final + note de TP) dépend de votre compréhension globale.



Les autres TD ... Et les TP ...

Les deux autres TD dépendent du TD1, ils sont la suite ... Les TP c'est différent, c'est pas le même problème qui est posé, mais tout le code que vous aurez fourni en TD vous sera utile.

3.1.3.1. Amélioration du code PHP

Voici un exemple de ce qu'il faut faire ... En gros, on a travaillé jusqu'à présent en mode brouillon, il faut nettoyer le code, le rendre de fait plus lisible. Donc vous devez le faire pour tout le code du projet !

Figure 3.9. Une partie du fichier `ajouter_produit.php` modifié

```

1
2
3 if (isset($_POST['submit'])) {
4     if (!hash_equals($_SESSION['csrf'], $_POST['csrf'])) die();
5
6     try {
7         $connection = new PDO($dsn, $username, $password, $options);
8
9         $nv_prod = array(
10             "code_produit" => $_POST['code_produit'],
11             "libelle"      => $_POST['libelle'],
12             "prix_unitaire" => $_POST['prix_unitaire']
13         );
14
15         $sql = sprintf(
16             "INSERT INTO %s (%s) values (%s)",
17             "Produit",
18             implode(" ", array_keys($nv_prod)),
19             ":" . implode(":", array_keys($nv_prod))
20         );
21
22         $statement = $connection->prepare($sql);
23         $statement->execute($nv_prod);
24     } catch(PDOException $error) {
25         echo $sql . "<br>" . $error->getMessage();
26     }
27 }
28 require "templates/header.php";
29
30 if (isset($_POST['submit']) && $statement) :
31     echo "<blockquote>" . escape($_POST['code_produit']) . " successfully added.</
blockquote>";
32 endif; ?>
33
34

```



Suppression de certaines balises ouvrantes / fermantes

L'idée quand il y en a trop, c'est d'utiliser un `echo` avec des concaténations.



Dans le cas d'utilisation d'un moteur de template

C'est plus facile, car tout le code html est dans un template, des fonctions PHP permettent d'écrire dans le template. C'est une architecture plus propre, et c'est d'ailleurs pour cela que les moteurs de templates ont été développés. De plus, on peut dans ce cas avoir une architecture MVC.

3.1.3.2. Utilisation d'Ajax

Retrouver l'adresse de livraison d'une commande. Deux listes liées entre elles. C'est un cas classique où Ajax semble inévitable. vous nommerez le fichier `adresse_livraison.php`, une bonne idée serait d'utiliser comme base de travail le fichier `commander.php` qui possède déjà des listes déroulantes.

Lisez donc l'article suivant ... Web 2.0, allez plus loin avec AJAX et XMLHttpRequest [<https://siddh.developpez.com/articles/ajax/>]

Figure 3.10. Formulaire principal

The screenshot shows a web browser window with the address bar displaying '127.0.0.1/magasin/magasi'. The page title is 'Système simple de gestion d'un magasin'. The main content area is divided into four light blue boxes, each with a description and a green button:

- Top-left: 'En utilisant ce lien vous pourrez ajouter un client dans la BDD.' with a button 'Ajouter un client'.
- Top-right: 'En utilisant ce lien vous pourrez ajouter un produit dans la BDD.' with a button 'Ajouter un produit'.
- Bottom-left: 'En utilisant ce lien vous pourrez commander un produit.' with a button 'Commander un produit'.
- Bottom-right: 'Retrouver l'adresse de livraison d'une commande.' with a button 'Adresse livraison'.

Figure 3.11. Adresse livraison 1/2

The screenshot shows the 'Adresse livraison' form within the same web browser window. The page title remains 'Système simple de gestion d'un magasin'. The form is titled 'Consulter l'adresse de livraison' and contains two dropdown menus:

- 'Nom du client' with the value 'Tudor' selected.
- 'Produit' with the value 'Choisir un produit' selected.

Below the dropdowns is a green 'Valider' button. On the left side of the form, there is a blue 'Retour' button.

Figure 3.12. Adresse livraison 2/2

← → ↺ 🏠 127.0.0.1/magasin/magas 90 % ... 📌 ⬇️ 🔍 Rechercher ☰ >> ☰

Système simple de gestion d'un magasin

L'adresse est : 14 rue des invalides

Consulter l'adresse de livraison

Nom du client
DERICK

Produit
MAUVAIS PRODUIT

Valider

Retour



C'est pas optimisé ...

C'est simplement histoire de le mettre en oeuvre, pas d'optimisation en effet, mais l'essentiel y est !

Démarche à suivre (3 fichiers à coder) :

- Lire l'article joint, puis réfléchissez à la méthode la plus rapide pour coder ce qui est demandé ! Oui, la plus rapide !
- Vous aurez besoin d'une requête SQL, vous permettant pour un client donné de récupérer la liste de ce qu'il a commandé.

Figure 3.13. La requête SQL (en cadeau)

```
// Requête pour récupérer tous les 'id_produit' et leur 'libelle'  
$sql_1 = "SELECT P.`id_produit`, P.`libelle` FROM `Commande` AS C, `Produit` AS P WHERE  
C.`id_client` = ". $_POST['id_client'] . " AND P.`id_produit` = C.`id_produit` ";
```

- Premier fichier : modifier le fichier header.php afin qu'il tienne compte du code Ajax ;
- Deuxième fichier : créer un fichier (ajaxadresse.php) réalisant la requête produisant le code du select. Donc requête SQL.
- Troisième fichier : copier le fichier commander.php en adresse_livraison.php puis modifier le ...
- Résumons : vous avez une documentation à lire, vous devez donc adapter ce code, ci-dessous une très grande partie du travail vous est donnée. Donc, il vous reste en tout et pour tout le début du fichier adresse_livraison.php à coder.



Est ce possible ?

Oui à condition de bien comprendre le tutoriel, et de bien comprendre le code fourni. C'est un travail de déduction, puis de codage de la partie manquante.

Figure 3.14. Une partie du fichier header.php (code Ajax)

```
1
2
3 <script type='text/javascript'>
4 function getXhr(){
5     var xhr = null;
6     if(window.XMLHttpRequest) // Firefox et autres
7         xhr = new XMLHttpRequest();
8     else if(window.ActiveXObject){ // Internet Explorer
9         try {
10             xhr = new ActiveXObject("Msxml2.XMLHTTP");
11         } catch (e) {
12             xhr = new ActiveXObject("Microsoft.XMLHTTP");
13         }
14     }
15     else { // XMLHttpRequest non supporté par le navigateur
16         alert("Votre navigateur ne supporte pas les objets XMLHttpRequest...");
17         xhr = false;
18     }
19     return xhr;
20 }
21
22 /**
23  * Méthode qui sera appelée sur le click du bouton
24  */
25 function go(){
26     var xhr = getXhr();
27     // On définit ce qu'on va faire quand on aura la réponse
28     xhr.onreadystatechange = function(){
29         // On ne fait quelque chose que si on a tout reçu et que le serveur est ok
30         if(xhr.readyState == 4 && xhr.status == 200){
31             leselect = xhr.responseText;
32             // On se sert de innerHTML pour rajouter les options à la liste
33             document.getElementById('id_produit').innerHTML = leselect;
34         }
35     }
36
37     // Ici on va voir comment faire du post
38     xhr.open("POST", "ajaxadresse.php", true);
39     // ne pas oublier ça pour le post
40     xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
41     // ne pas oublier de poster les arguments
42     // ici, l'id du client
43     sel = document.getElementById('id_client');
44     id_client = sel.options[sel.selectedIndex].value;
45     xhr.send("id_client="+id_client);
46 }
47 </script>
48
49
```

Figure 3.15. Fichier ajaxadresse.php

```

1
2
3 <?php
4 require "../config.php";
5 require "../common.php";
6
7 try {
8     $connection = new PDO($dsn, $username, $password, $options);
9
10    // Requête pour récupérer tous les 'id_produit' et leur 'libelle'
11    $sql_1 = "SELECT P.`id_produit`, P.`libelle` FROM `Commande` AS C, `Produit` AS P
12    WHERE C.`id_client` = ". $_POST['id_client'] . " AND P.`id_produit` = C.`id_produit` ";
13
14    $statement_1 = $connection->prepare($sql_1);
15    $statement_1->execute();
16    $result_1 = $statement_1->fetchAll();
17
18    } catch(PDOException $error) {
19        echo $sql_1 . "<br>" . $error->getMessage();
20    }
21
22    echo "<select name='id_produit'>";
23    if ($result_1 && $statement_1->rowCount() > 0)
24        foreach ($result_1 as $row_1) :
25            echo "<option value='". escape($row_1['id_produit']). "'>".
26            escape($row_1['libelle']). "</option>";
27        endforeach;
28    echo "</select>";
29
30    ?>

```

Figure 3.16. Une partie du fichier `adresse_livraison.php`

```
1
2
3 <div class="grid-x grid-padding-x">
4     <div class="large-3 medium-3 small-12 cell"></div>
5     <div class="large-6 medium-6 small-12 cell">
6         <h5>Consulter l'adresse de livraison</h5>
7         <form method="post">
8             <input name="csrf" type="hidden" value="<?php echo
escape($_SESSION['csrf']); ?>">
9             <div class="grid-x grid-padding-x">
10                 <div class="large-12 cell">
11                     <label>Nom du client</label>
12                     <select name='id_client' id='id_client' onchange='go()'>
13                         <?php
14                             if ($result_0 && $statement_0->rowCount() > 0)
15                                 foreach ($result_0 as $row_0) :
16                                     echo "<option value='".
escape($row_0['id_client']). "'>". escape($row_0['nom']) . "</option>";
17                                 endforeach;
18                             ?>
19                         </select>
20                     </div>
21                     <div class="large-12 cell">
22                         <label>Produit</label>
23                         <div id='id_produit' style='display:inline'>
24                             <select name='id_produit'>
25                                 <option value='-1'>Choisir un produit</option>
26                             </select>
27                         </div>
28                     </div>
29
30                     <div class="large-12 cell">
31                         <div class="float-
right"><input type="submit" name="submit" value="Valider" class="success button"/></div>
32                         </div>
33                     </div>
34                 </form>
35             </div>
36             <div class="large-3 medium-3 small-12 cell"></div>
37         </div>
38
39     <div class="grid-x grid-padding-x">
40         <div class="large-12 medium-12 small-12 cell"></div>
41         <a href="index.php" class="Alert button">Retour</a>
42     </div>
43 </div>
44
45
```

3.1.3.3. Ce qui reste à faire !

Si vous avez envie de réussir à atteindre un certain niveau :

- Développer les interfaces de suppression des clients, produits ...
- On a pas du tout modifié le thème par défaut de Foundation et franchement c'est pas terrible ! Alors prenez le temps de lire les docs sur le site officiel de Foundation. Puis commencer par exemple à changer les couleurs en modifiant / rajoutant du code `SCSS` dans le fichier `_settings.scss` !



Pour quelles raisons, je chercherais à faire ce qui est demandé ici ?

Pour en apprendre le plus possible, afin d'être opérationnel pour le TP à rallonge. C'est dans l'esprit de gagner du temps.

3.1.4. TP1 : *Le Grand Hôtel* - MCD et requêtes SQL

Le Grand Hôtel de la plage de Testraou à Perros GUIREC cherche à moderniser son système de gestion de chambre. L'hôtel possède actuellement 45 chambres. La direction souhaite un logiciel de gestion et de réservation en ligne.



N'ont ils pas déjà un site web et un formulaire de réservation ?

Oui, mais toute la partie intelligente du site n'est pas développée. L'interface web de réservation (Site officiel du *Le Grand Hôtel* [<http://www.grand-hotel-perros-guirec.com>]) est sous traitée à un intervenant extérieur.

Vision globale des objectifs :

- Réserver une chambre en ligne ;
- Calculer le coût de celle-ci, en fonction de la période choisie (creuse ou haute saison), du type de chambre (standard, supérieure, exceptionnelle (suite)) et du petit déjeuner inclus ou non ;
- Proposer la facture directement au client en fonction de ses choix.

En partie administration, le gestionnaire pourra :

- Attribuer une remise à un client sur un séjour ;
- Entrer, modifier, supprimer via différentes interfaces : les comptes clients, les tarifs (modifier les périodes), les chambres, les réservations ;
- Supprimer un compte client, dans ce cas il disparaîtra de toutes les tables dans lesquelles il a été référencé (Réservation, Lignefacture, Facture et Compte ... *C.F. En première analyse le MCD suivant ci-dessous*) ;
- Interdire à la location une chambre (utilisable à faux).

Quelques règles :

- Un client doit créer un compte sur l'interface afin de valider une réservation ;
- Le client peut réserver une chambre tout de suite après son enregistrement ;
- Si un client réserve, il doit renseigner une date de début et une date de fin de séjour, le type de chambre (standard, supérieure, exceptionnelle), le nombre de personnes adultes (nombre de lits) et s'il souhaite prendre un supplément petit déjeuner ;
- Un client peut réserver une chambre pour l'année en cours et déborder éventuellement sur la suivante, mais pas plus ;

Déroulement.

- Travailler par groupe de 3 étudiants (mini 2 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 3 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).

Les indispensables.

- Une machine virtuelle bien utile [<http://partage.rt-iut.re/rtcloud/Machines-Virtuelles/VM-Linux/devuan/>]
- Lien pour télécharger AnalyseSI [<https://launchpad.net/analyses/+/download>]

- Codes sources et début d'analyse MCD (v4)

[<https://drive.google.com/drive/folders/1B3o3UTt7Btufo8B4mGrZyo-C6MiiNCYd?usp=sharing>]



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TP, la suite (examen final + note de TP) dépend de votre compréhension globale.



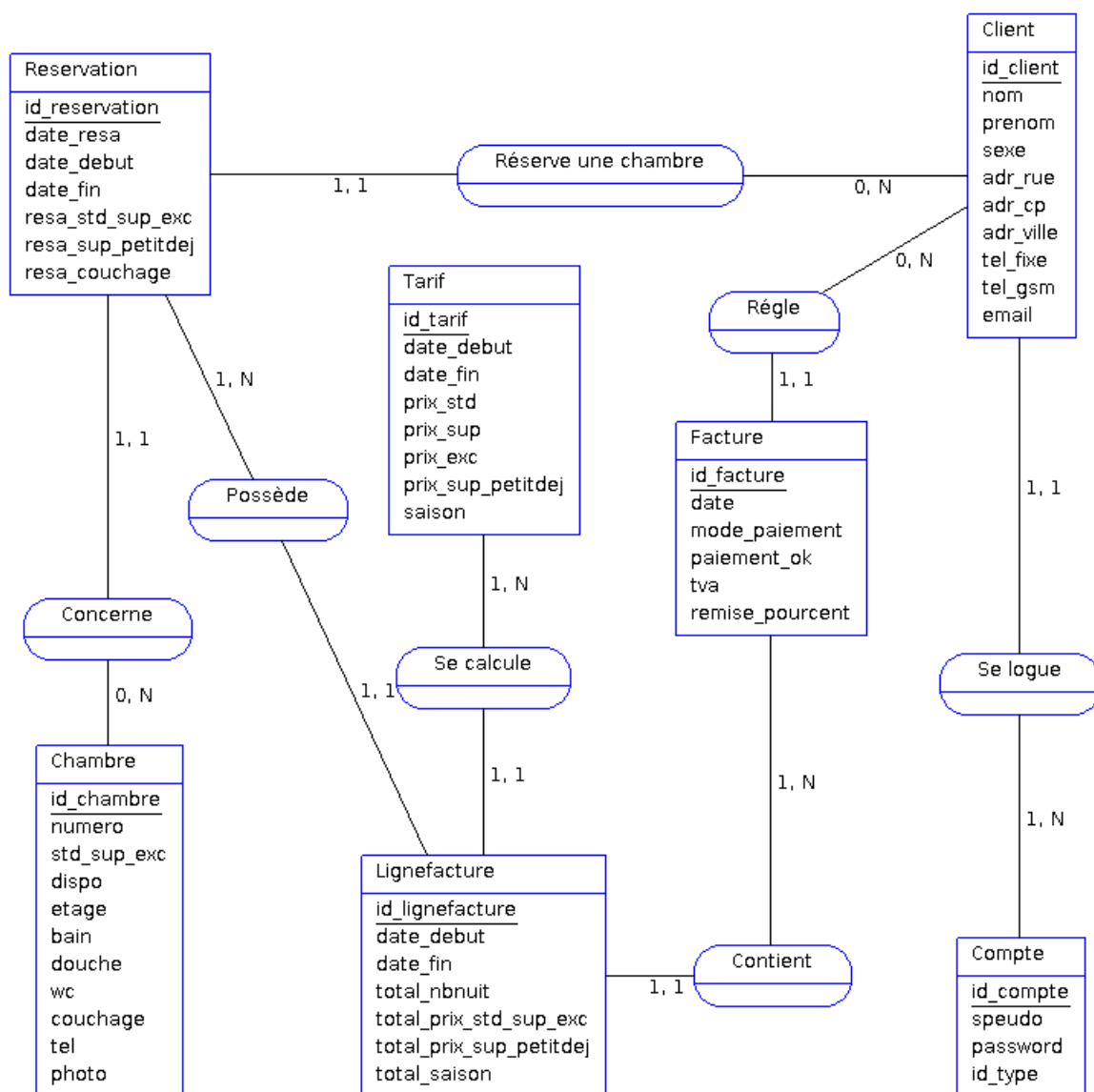
Et les autres TP ...

C'est une démarche de projet, dans ce cadre les TP sont tous liés les uns aux autres. C'est pour cela, qu'il vous faut avancer correctement, pour ne pas être à un certain moment distancé.

3.1.4.1. Modifier le modèle MCD (AnalyseSI)

Le modèle actuel est disponible sur Moodle. Vous devez donc le modifier, en tenant compte des remarques qui suivent. Vous utiliserez AnalyseSI !

Figure 3.17. En première analyse le MCD suivant



Règles générales liées aux cardinalités

Réserve une chambre	Un client effectue au moins 0 et au plus plusieurs réservations. Une réservation est effectuée par au moins 1 et au plus 1 client.
Se logue	Un client et au plus 1 peut se loguer sur son compte. Un compte est concerné par au moins 1 et au plus 1 client.
Règle	Un client règle 0 à N facture. Une facture est réglée par au moins 1 et au plus un client.
Concerne	Une réservation concerne au moins 1 et au plus 1 chambre. Une chambre est concernée par au moins 0 et au plus plusieurs réservations (à des périodes ou dates différentes).
Possède	Une réservation possède au moins une ligne facture et au plus N. Une ligne facture est liée à au moins 1 et au plus 1 client.
Se calcule	Une ligne facture et 1 seule ligne facture se calcule en fonction d'un tarif. Un tarif est utilisé pour calculer 1 et au plus N ligne facture.
Contient	Une facture contient au moins 1 et au plus N "lignes de facture". Une ligne de facture est contenue dans au moins 1 et au plus 1 facture.

Vous devez modifier le modèle suivant afin qu'il tienne compte du fait qu'un client peut réserver aussi pour un enfant ou des enfants (moins de 13 ans).

Ceci signifie que dans la table *Reservation* le champ *resa_couchage* doit évoluer en deux champs *resa_nb_adulte* et *resa_nb_enfant*. Dans la table *Chambre* le champ *couchage* doit changer en *nb_lits* (le terme est plus approprié, comprendre lit pour adulte(s)), les lits pour enfants ne sont pas utilisés pour choisir la chambre car ils sont amovibles et peuvent être rajoutés à la dernière minute.

Dans la table *Lignefacture*, il faut rajouter un champ *total_prix_sup_enfant* afin d'en tenir compte lors de la facturation.

Dans la table *Tarif*, il faut rajouter un champ *prix_sup_enfant* pour la cohérence du modèle.

Lorsque vous aurez terminé l'analyse avec AnalyseSI vous devrez générer le code SQL du projet. Puis en utilisant phpmyadmin l'importer et ainsi obtenir une base de données de fonctionnelle.



Si vous utilisez la machine virtuelle proposée sur rt-cloud ...

Vous avez dans le *home directory* un script permettant de créer un compte MySQL pour votre projet ...

3.1.4.2. Définir des requêtes SQL

Maintenant, vous devez entrer des données dans vos tables, afin d'alimenter le modèle pour pouvoir faire des requêtes. Pour cela, vous utiliserez tous des périodes et des tarifs identiques, les mêmes clients, les mêmes comptes ... Bref, il serait fastidieux de vous demander d'inventer des données. De plus ce serait très difficile de déboguer vos requêtes ou codes PHP si des données différentes étaient utilisées. Sur Moodle vous avez le fichier *datas.tar.gz* comprenant du code SQL permettant d'alimenter vos tables.



Beaucoup de code SQL, dois-je garder une copie propre ?

Oui, une copie sans avoir utilisé le fichier *datas.tar.gz*, afin de pouvoir revenir *facilement* au point de départ.

Soient les requêtes suivantes à coder (SQL) :

- La liste de tous les clients avec toutes les informations (adresse, numéro de téléphone ...) ;
- La liste de tous les administrateurs du site ;
- Le nombre de clients inscrits dans la base ;
- Le nombre de clients ayant effectués une réservation ;
- Afficher les informations complètes sur les réservations des clients (si un client a effectué plusieurs réservations, il doit apparaître plusieurs fois) ;
- Afficher les informations complètes sur les réservations des clients entre les dates '2018-06-11' et '2018-12-31' (si un client a effectué plusieurs réservations, il doit apparaître plusieurs fois) ;

3.1.5. TP2 : Foundation intégration

Vous avez défini le modèle avec AnalyseSI puis généré le code SQL. Maintenant vous allez créer la page de garde du site de réservation en ligne pour *Le Grand Hôtel*. Vous intégrerez le code de *datepicker* afin que l'internaute puisse choisir la période de réservation.

Figure 3.18. Exemple de page de garde 1/2

localhost:3000/# 67 % Rechercher

LE GRAND HÔTEL

Visite Guidée Documentation Informations légales À Propos

Bienvenue au Grand Hôtel

Disponibilités
Choisissez vos dates :

Arrivée : 21/01/2019 Départ : 30/01/2019 Nuit(s) : 9

Petit déjeuner :
☒ Oui
☐ Non

Nombre de lit : 2

Chambre : Supérieure

Je réserve !

Petit déjeuner

Informations générales

L'hôtel vous propose 45 chambres conçues pour le bonheur de toutes les familles et de tous les couples. Un panel de services vous est proposé afin de rendre votre séjour le plus agréable possible.

Des services

- wi-fi gratuit dans tout l'hôtel
- journaux/presse gratuits à la réception
- bar avec rafraîchissements sans alcool (chauds ou froids) offerts de 15h à 17h30
- vélos électriques à disposition gratuitement sur réservation
- parking privé gratuit
- ascenseur
- prêt de lits bébés sur demande

Des divertissements et animations

- salon avec billard vue sur mer
- salle de jeux avec babyfoot, jeux de société, jeux vidéo, tennis
- accès gratuit aux séances de sport, yoga, Pilates avec coach sportif



Nombre de lits ?

Vous remarquerez que le nombre de lits ne tient pas compte du fait qu'il peut y avoir des adultes et des enfants (moins de 13 ans) et qu'ils ne sont pas facturés de la même façon.

Figure 3.19. Exemple de page de garde 2/2

localhost:3000/# 67 %

LE GRAND HÔTEL

Visite Guidée

Documentation

Informations légales

À Propos

Arrivée : 21/01/2019

Départ : 22/01/2019

Nuit(s) : 1

Petit déjeuner :

☒ Oui

☐ Non

Nombre de lit

1

Chambre

Standard

Je réserve !

Objectifs :

- Réaliser la page principale du site puis les autres pages (Visite Guidée, Documentation, Informations légales, A Propos (Contact, Tarifs, Espace membre)) ;
- Intégrer le code de *datepicker* ;
- Disposer intelligemment les blocks afin de proposer une interface conviviale ;
- Modifier le thème par défaut que propose Foundation ;
- Vous avez modifié le modèle MCD afin de tenir compte de la remarque ci-dessus (adultes / enfants), faites en sorte que l'interface web en tienne compte ;

- Intégrer au site une carte *google map*. Foundation propose très certainement du code pour le faire facilement ;
- Enfin soyez créatif et faites mieux que ce qui est proposé. Vous trouverez sur internet toutes les informations nécessaires, images et site web, menu responsive à modifier :
- Site officiel du *Le Grand Hôtel* [<http://www.grand-hotel-perros-guirec.com>]
- TripAdvisor *Le Grand Hôtel* [https://www.tripadvisor.fr/Hotel_Review-g207340-d1024359-Reviews-Le_Grand_Hotel-Perros_Guirec_Cotes_d_Armor_Brittany.html]
- LogiTravel *Le Grand Hôtel* [<https://www.logitravel.fr/hotels/europe/france/perros-guirec/le-grand-hotel-trestraou-17862907.html>]
- Expédia.fr *Le Grand Hôtel* [<https://www.expedia.fr/Cotes-DArmor-Hotel-Grand-Hotel-De-Trestraou.h23478575.Description-Hotel>]
- Foundation - Top Bar with Responsive Toggle and Animation (CSS/Float Example) [<https://codepen.io/IamManchanda/pen/KmepBg>]
- Foundation 6 Megamenu Test [<https://codepen.io/madsciencepro/pen/WRwyvP>]
- Foundation 6 Menu Examples [<https://codepen.io/SitePoint/full/bEYPmg/>]
- Foundation 6 Top Bar [codepen.io/shoaibik/pen/wGORZY]

Déroulement.

- Travailler par groupe de 3 étudiants (mini 2 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 3 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TP, la suite (examen final + note de TP) dépend de votre compréhension globale.



Et les autres TP ...

C'est une démarche de projet, dans ce cadre les TP sont tous liés les uns aux autres. C'est pour cela, qu'il vous faut avancer correctement, pour ne pas être à un certain moment distancé.

3.1.5.1. Quelques points techniques

Vous devez organiser votre projet du point de vue Foundation :

Quelques suggestions et précisions sur le travail à fournir :

- Vous utiliserez un fichier SCSS particulier (`_custom.scss`) qui se trouvera dans le répertoire `scss/` vous y coderez le plus de code possible (parfois ce ne sera pas possible, il vous faudra modifier directement l'un des fichiers `_settings.scss` ou `app.scss`). Il faudra l'intégrer via (en fin du fichier `app.scss`) :

```
1
2
3  @import 'custom';
4
5
```



Dès que vous modifiez un fichier autre que `_custom.scss`

Indiquez par un commentaire facile à retrouver ce que vous avez modifié, toujours commenter l'original, ne pas effacer ...

- Le menu doit être responsive, il doit y avoir le logo de l'hôtel (disponible sur Moodle), affichage correct pour tous les types d'écran, le menu doit être *sticky* (se plaquer en haut quand on descend dans la page), il doit aussi changer en fonction de la taille de l'écran. Enfin, vous serez amené à modifier les *médias queries* directement dans le fichier `_custom.scss` (2. Breakpoints) ;
- Intégration de *datepicker*. J'ai modifié le code (`js/datepicker_custom.js`) afin de tenir compte du calcul du nombre de nuit. Voici en gros ce qu'il faut rajouter dans la page `index.html` :

```
<!doctype html>
<html class="no-js" lang="en">
  <head>
    <meta ....

    <link rel='stylesheet' href='css/foundation-datepicker.min.css'>

  </head>
  <body>

    Arrivée : <input type="text" class="span2" value="" id="dpd1">
    Départ : <input type="text" class="span2" value="" id="dpd2">

    Nuité(s) : <input type="text" class="span2" value="1" id="dpd3">

    <script src='js/foundation-datepicker.min.js'></script>
    <script src='js/locales/foundation-datepicker.fr.js'></script>
    <script src="js/datepicker_custom.js"></script>

  </body>
</html>
```

Une archive est fournie sur Moodle fichier `datepicker_code_Moodle.tar.gz` ;

- Modifier la police par défaut (police pour le web récupérée sur internet), choisissez la votre ! Pour cela, créer un répertoire `fonts/`, mettez y tout ce qu'il faut pour que la police fonctionne, le fichier `css` qui y fera référence sera dans le répertoire `css/`, intégrer ensuite la police à votre fichier `_custom.scss` via :

```
@import url(fontproximanova.css);
```

- Vous devez avoir un carrousel responsive, avec des images et/ou des textes ;
- La page principale doit contenir un haut de page, header avec image ou texte, un bas de page footer contenant des blocks. Vous devez créer des blocks dans le corps `body` des pages et être capable d'expliquer comment ils s'agencent en fonction de la taille de l'écran ;



Je récupère du code ici ou là ...

Oui, mais attention à la version du code que vous allez récupérer, toujours du code pour Foundation 6.5.1 ou version supérieure !



Et si on a pas terminé dans les temps ...

Vous finirez chez vous ou au fil des TP, parfois après avoir mieux compris certaines choses on réalise de grandes modifications très rapidement. Mais vous devez avoir fait tout ce qui est demandé ici !

3.1.6. TP3 : Gestion des comptes utilisateur

Vous devez maintenant coder la gestion des comptes utilisateurs. La réservation proprement dite sera développée dans le TP suivant.

Deux types de comptes :

- Les comptes client : lorsque la personne valide sa réservation elle se retrouve sur une page lui demandant ses identifiants (login/password) nécessaires à son authentification sur l'interface. Un lien sera également disponible afin qu'elle puisse s'enregistrer. Dans ce cas, des informations à la création d'un compte sur l'interface lui seront alors demandées. Ici pas de confirmation par envoi de mail, car nous n'avons pas la main sur cet aspect (messagerie) du réseau R&T ;
- Les comptes administrateur : vous devez réaliser la gestion des comptes administrateur. Les administrateurs vous l'aurez compris ont des droits particuliers sur l'interface. Pour cela vous devrez impérativement utiliser le mécanisme des *sessions* proposé par PHP ;

Méthode de travail :

- Dans un premier temps vous allez modifier le champ `input` indiquant le nombre de nuits, car ce champ est modifiable. Or, même si c'est pas très difficile à faire, il vaut mieux que l'internaute ne puisse pas le modifier, car lors du choix des dates ce champ est mis à jour automatiquement (comment je fais ça ? *google* -> *html input disabled*) ;
- Après avoir validé le formulaire de réservation, le client arrive sur une page lui proposant de se loguer ou de créer un compte sur l'interface. Donc un nouveau formulaire, attention il est hors de question de perdre les informations précédentes après validation du formulaire de login. Une bonne idée serait de mettre des champs `hidden` contenant les dates pour la réservation et le nombre de nuits, comme cela elles seront re-postées ;
- Maintenant, si l'internaute doit créer un compte sur l'interface, il faut aussi coder des champs `hidden` contenant les dates pour la réservation et le nombre de nuits. Vous coderez le formulaire d'enregistrement pour un nouveau client (tous les champs du modèle MCD doivent être renseignés). Pour faire simple, il ne faut pas changer de page, donc en cliquant sur le lien un formulaire apparaît ... Comment je fais une chose pareille ? Avec un `Dropdown` [<https://foundation.zurb.com/sites/docs/dropdown.html>] par exemple !
- Dans le menu *responsive*, vous avez créé un lien : Espace membre. Cet espace membre permet à l'administrateur de se loguer (tout comme un client pourrait le faire), donc un formulaire identique au précédent (peut être le même d'ailleurs) ;
- Vous devez coder l'affichage de tous les utilisateurs (partie administration). Une fois logué l'administrateur peut consulter la base de tous les clients, il peut en supprimer, en rajouter, modifier les informations les concernant ;
- Vous coderez également (sinon ça ne fonctionnera pas) en PHP une *classe* qui serait logique d'appeler `Client` proposant des méthodes permettant :
 - D'ajouter un client ;
 - Supprimer un client ;
 - La mise à jour d'un client ;



Est ce tout ?

Par la suite vous verrez et vous rajouterez les méthodes nécessaires.

Déroulement.

- Travailler par groupe de 3 étudiants (mini 2 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 3 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TP, la suite (examen final + note de TP) dépend de votre compréhension globale.



Et les autres TP ...

C'est une démarche de projet, dans ce cadre les TP sont tous liés les uns aux autres. C'est pour cela, qu'il vous faut avancer correctement, pour ne pas être à un certain moment distancé.

3.1.6.1. Quelques points techniques

Vous devez utiliser abide (du projet Foundation) afin de permettre de vérifier les champs de vos formulaires.

abide ? Vous avez dit abide :

- Faites un copié / collé d'un exemple simple dans votre page html. Comme cela vous pourrez vérifier facilement si ça fonctionne, après vous pourrez modifier le code. Exemple (c'est pas ce qu'il y a de plus simple, mais ça fonctionne !) :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Foundation Template</title>
    <meta name = "viewport" content = "width = device-width, initial-scale = 1">

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/foundation-sites@6.5.1/dist/
css/foundation.min.css">

    <!-- Compressed JavaScript -->
    <script src = "https://cdnjs.cloudflare.com/ajax/libs/foundation/6.0.1/js/vendor/
jquery.min.js"></script>

    <script src="https://cdn.jsdelivr.net/npm/foundation-sites@6.5.1/dist/js/
foundation.min.js"></script>
  </head>

  <body>
    <h2>Example of Abide</h2>
    <form data-abide novalidate>
      <div data-abide-error class = "alert callout" style = "display: none;">
        <p><i class = "fi-alert"></i> There are some errors in your form.</p>
      </div>

      <div class = "row">
        <div class = "small-12 columns">
          <label>Name
            <input type = "text" placeholder = "Name" required pattern = "[a-zA-Z]+">
            <span class = "form-error">Fill the Correct box</span>
          </label>

          <label>Email
            <input type = "text" placeholder = "abd@email.com" pattern = "email">
          </label>
        </div>
      </div>

      <div class = "row">
        <div class = "medium-4 columns">
          <fieldset>
            <label>Gender</label>
```



```

<input type = "radio" name = "pockets" value = "Male">
<label>Male</label>

<input type = "radio" name = "pockets" value = "Female">
<label>Female</label>

<input type = "radio" name = "pockets" value = "Other">
<label>Other</label>
</fieldset>

<label>Input Label
  <select id = "select" required>
    <option value = ""></option>
    <option value = "volvo">Volvo</option>
    <option value = "saab">Saab</option>
    <option value = "mercedes">Mercedes</option>
    <option value = "audi">Audi</option>
  </select>
</label>
</div>
</div>

<div class = "row">
  <fieldset class = "large-6 columns">
    <button class = "button" type = "submit" value = "Submit">Submit</button>
    <button class = "button" type = "reset" value = "Reset">Reset</button>
  </fieldset>
</div>
</form>

<script>
$(document).ready(function() {
  $(document).foundation();
})
</script>
</body>
</html>

```



Je trouve cela où ?

C'est ici : Abide démo [https://www.tutorialspoint.com/foundation/abide_demo.htm] ...
 D'autres liens : Un exemple avec Foundation 5.5.3 (identique à la dernière version) [<https://foundation.zurb.com/sites/docs/v/5.5.3/components/abide.html>], Doc officielle Foundation zurb [<https://foundation.zurb.com/sites/docs/abide.html>] ...

- Le mot de passe doit être crypté. Pour cela, il faut utiliser une fonction PHP adéquate `sha1 ()`. Comment vérifier ensuite que le mot de passe est le bon ? C'est simple, on compare les cryptés, sachant que l'on ne va pas décrypter le mot de passe car c'est mission impossible. Soit l'exemple de code ci-dessous :

```

<?php
$mdp = sha1($_POST['mot_de_passe']);
?>

```

Mais c'est pas encore suffisamment solide, car si un pirate a réussi à casser `sha1 ()`, dans ce cas c'est perdu. Il y a une combine qui consiste à rajouter une chaîne compliquée au mot de passe :

```
<?php

// on définit une chaîne assez complexe avec
// des caractères en tout genre ...
define('PREFIXE_SHA1', 'p8%B;Qdf78');

// On récupère le mot de passe du formulaire (méthode POST)
$mdp = $_POST['mot_de_passe'];

// On concatène le mot de passe crypté avec la chaîne PREFIXE_SHA1
$mdp_shal = sha1(PREFIXE_SHA1.$mdp);

?>
```

3.1.7. TP4 : La réservation

Vous devez maintenant coder la réservation des chambres. C'est la partie la plus conséquente de cette série de TP. Il vous manque quelques règles de gestion avant de commencer le travail.

Règles liées à la réservation d'un type de chambre en particulier :

- La qualité globale de la chambre change suivant son classement (standard, supérieure, exceptionnelle (suite)) ainsi que le nombre de lits qu'elle contient :
 - Une chambre standard possède 1 lit simple ;
 - Une chambre supérieure possède 1 lit double ;
 - Une chambre exceptionnelle (suite) possède 2 lits doubles et 1 lit simple.
- Dans certains cas, une chambre peut avoir un ou plusieurs lits simples en plus (que l'on peut rajouter vue qu'il y a de la place disponible), dans ce cas de figure l'information est fournie par le nombre de lits (nb_lits). Exemples :
 - Chambre standard, le nombre de lit est de 3, cela signifie qu'il y a 1 lit simple + 2 lits simples (3 adultes) ;
 - Chambre supérieure, le nombre de lit est de 2, cela signifie qu'il y a 1 lit double + 1 lit simple (3 adultes) ;
 - Chambre exceptionnelle (suite), le nombre de lit est de 4, cela signifie qu'il y a 2 lits doubles + 2 lits simples (6 adultes).
- L'internaute choisit donc le type de chambre (standard, supérieure, exceptionnelle (suite)) sachant que cela correspond aussi, en plus de la qualité globale de la chambre, au fait qu'il y ait un ou des lits simples ou doubles. Tous les lits supplémentaires sont des lits simples. Une personne seule a la possibilité de choisir une chambre supérieure, voire exceptionnelle. Dans ce cas elle dormira dans un lit double, le coût en sera de fait plus élevé.
- L'internaute doit pouvoir choisir le nombre d'adulte et le nombre d'enfant pour une réservation, mais c'est limité :
 - Pas plus de 7 adultes ;
 - Pas plus de 4 enfants.
- Il doit aussi pouvoir choisir le type de chambre :
 - Standard ;
 - Supérieure ;
 - Exceptionnelle (suite).
- Il doit aussi pouvoir choisir le nombre d'enfants (<13 ans). Un supplément par enfant est ajouté à la facture globale (aussi à l'entrée concernée dans la table Lignefacture).

Ligne facture, facture c'est quoi ?

- Une ligne facture correspond à une partie ou éventuellement à la totalité d'une réservation durant une période définie dans la table 'Tarif'. Si une réservation est étalée sur plusieurs périodes tarifaires différentes, il y aura autant de ligne dans la table 'Lignefacture' que de périodes ;
- La facture reprend les lignes de la table 'Lignefacture' ;
- La facture tient compte de la TVA, du paiement réel de la facture, d'une éventuelle remise, du mode de paiement. Bien entendu la date du paiement est indiquée dans le champ date.

- Donc pour créer une facture *réelle* il faut faire jouer au travers de requêtes les tables 'Lignefacture' et 'Facture'.

Règles liées aux tarifs des chambres et à la ligne facture :

- Les tarifs sont consus par période (creuses ou non) et pour les chambres standards, supérieures, exceptionnelles (suites). Si le client a choisi l'option petit déjeuner, alors un supplément est à appliquer. Il faudra multiplier le supplement par le nombre de jours du séjour ;
- De même pour le supplement enfant, à ceci prêt qu'il faudra aussi considérer le nombre d'enfant ;
- Pour le supplement qui concerne le nombre de lits ce sera la même chose, il faudra aussi considérer le nombre de lit dans le calcul final.

Déroulement.

- Travailler par groupe de 3 étudiants (mini 2 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 3 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TP, la suite (examen final + note de TP) dépend de votre compréhension globale.



Et les autres TP ...

C'est une démarche de projet, dans ce cadre les TP sont tous liés les uns aux autres. C'est pour cela, qu'il vous faut avancer correctement, pour ne pas être à un certain moment distancé.

3.1.7.1. L'agorithme permettant de renseigner la table 'Lignefacture'

Cet algorithme vous est fourni, mais il est assez compliqué. Vous devez vous l'approprier en le faisant *tourner* pour des réservations prédéfinies. Vous avez les requêtes sur Moodle (il vous faut des tarifs et des réservations en exemple).

La période de réservation souhaitée par l'internaute chevauche éventuellement une ou des périodes comportant des tarifs différents. C'est pour cette raison, qu'il faut définir précisément le tarif d'une ligne de la table 'Lignefacture'. Un client peut dans ce cadre avoir plusieurs lignes dans la table 'Lignefacture' pour une seule réservation.

Pour une période où le client a réservé, il peut y avoir différentes tarifications suivant qu'il passe en basse ou haute saison. A ce propos, l'information 'saison' est là simplement pour indiquer un changement important dans le tarif, car les tarifs sont modulables finement, et l'internaute peut se demander pourqu'elles raisons il y a une si grande différence de prix, il lira alors sur la facture qu'il est en haute saison.

```
dx = (date de début de la réservation)
dy = (date de fin de la réservation)
```

```
myArrayOf_Id_tarif = [[R1 retourne un tableau contenant tous les id_tarif qui sont concernés
par la période de réservation de l'internaute. Un chevauchement sur l'année suivante est
possible, mais uniquement une année. En d'autre terme, il n'est pas possible de réserver plus
de 2 ans moins 1 jour. Attention, ce tableau est parcouru en prenant la dernière période en
premier. ]]
```

```
POUR (toutes la valeurs du tableau myArrayOf_Id_tarif) FAIRE
```

```

d0 = [[[R2 date de début d'une période de la table 'Tarif' (pour l'id_tarif courant) ]]]
d1 = [[[R3 date de fin d'une période de la table 'Tarif' (pour l'id_tarif courant) ]]]

total_prix_std_sup_exc = [[[R4 le tarif de la chambre pour la période considérée (donc
pour l'id_tarif courant) ]]]

total_prix_sup_lits = [[[R5 (le tarif du supplément lits pour la période considérée (donc
pour l'id_tarif courant)) ]]]

total_prix_sup_enfant = [[[R6 (le tarif du supplément enfant pour la période considérée
(donc pour l'id_tarif courant)) * resa_nb_enfant ]]]

total_prix_sup_petitdej = [[[R7 le tarif du supplément petit déjeuner pour la période
considérée (donc pour l'id_tarif courant) ]]]

total_saison = [[[R8 l'information en saison ou hors saison ]]]

SI (dx != dy) ALORS
  SI (dx >= d0) ALORS
    nbnuit1 = dy - dx;

    total_prix_std_sup_exc = total_prix_std_sup_exc * nbnuit1;
    total_prix_sup_enfant = total_prix_sup_enfant * nbnuit1 * nb_enfant;
    total_prix_sup_petitdej = total_prix_sup_petitdej * nbnuit1 * nb_adulte;
    total_saison = total_saison * nbnuit1;

    [[[R9 INSERER les valeurs total_prix_std_sup_exc, total_prix_sup_enfant,
total_prix_sup_petitdej et total_saison dans la table `Lignefacture` ]]]

    // On se déplace dans l'interval de date en réduisant la période à examiner.
    dy = dx;

  SINON
    SI (dy >= d0) ALORS
      nbnuit2 = dy - d0;

      total_prix_std_sup_exc = total_prix_std_sup_exc * nbnuit2;
      total_prix_sup_enfant = total_prix_sup_enfant * nbnuit2 * nb_enfant;
      total_prix_sup_petitdej = total_prix_sup_petitdej * nbnuit2 * nb_adulte;
      total_saison = total_saison * nbnuit2;

      [[[R10 INSERER les valeurs total_prix_std_sup_exc, total_prix_sup_enfant,
total_prix_sup_petitdej et total_saison dans la table `Lignefacture` ]]]

      // On se déplace dans l'interval de date en réduisant la période à examiner.
      dy = d0;
    FIN SI
  FIN SI
FIN SI
FIN POUR

```



C'est quoi '[[[Rx ...]]]' ?

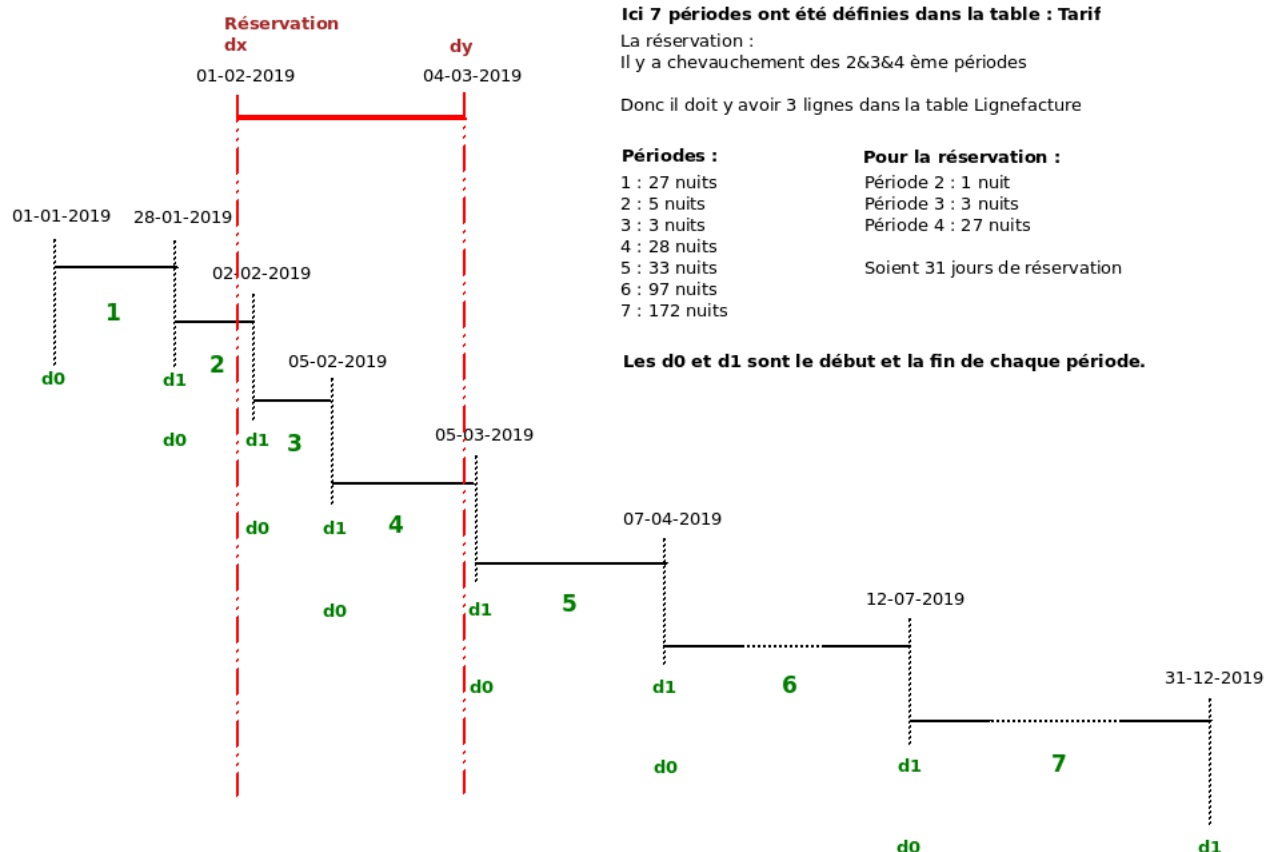
Cela signifie que vous devez faire une requête SQL pour obtenir ce qui est décrit ...



Il manque quelque chose !

Le total_prix_sup_lits n'est pas bon. En effet, il ne tient pas compte du calcul du nombre de lits décrit au tout début de ce TP.

Figure 3.20.



Cas d'école d'une réservation type

3.1.7.1.1. Votre travail

Mais pour le choix de la chambre, je fais comment ? Ce sera pour le TP suivant !

Méthode de travail :

- Vous devez comprendre l'algorithme proposé ! Ceci est très important pour la suite, vous avez aussi les requêtes permettant d'alimenter les tables en valeurs de fournies sur Moodle (fichier `requetes_mysql_alimentant_la_BDD.tar.gz`). Vous devez les utiliser ;
- Vous devez modifier l'algorithme afin qu'il ne tienne compte du calcul du nombre de lits décrit au tout début de ce TP ;
- Vous devez coder toutes les requêtes R1 à R10 SQL dont la description est fournie. Pour les mettre au point utiliser phpmyadmin ;
- Vous devez coder dans une classe PHP 'Reservation' toutes les méthodes nécessaires au bon fonctionnement de la partie réservation.
- Vous devez intégrer votre travail à l'interface web afin que cette partie fonctionne correctement.

Les requêtes Rx informations complémentaires

- R1 : Ce qu'il faut comprendre ici, c'est que la table 'Tarif' contient tous les tarifs de toutes les périodes depuis la mise en ligne de l'application de réservation de chambre d'hôtel en ligne. C'est pourquoi, pour éviter de faire tourner l'algorithme sur des années antérieures à l'année de réservation, il faut réduire le champ ... Vous

avez dans le langage SQL la possibilité d'extraire d'une date l'année, le mois ou le jour. Regarder donc du côté de `extract(...)`.

- R2 : vous devez récupérer la date de début de la période (donc pour l'id_tarif courant) ;
- R3 : vous devez récupérer la date de fin de la période (donc pour l'id_tarif courant) ;
- R4 & R5 : ici vous devez faire un calcul (dans votre code PHP) - en respectant les règles définies au début de ce TP - afin de définir correctement le `total_prix_std_sup_exc` et le `total_prix_sup_lits` (donc pour l'id_tarif courant). Il y a aussi de requêtes SQL à coder ;
- R6 : pour `total_prix_sup_enfant` vous devez penser à multiplier le `prix_sup_enfant` (résultat de votre requête) par le nombre d'enfant (donc pour l'id_tarif courant) ;
- R7 : pour `total_prix_sup_petitdej` vous devez penser à multiplier le `prix_sup_petitdej` (résultat de votre requête) par le nombre d'adulte (donc pour l'id_tarif courant) ;
- R8 : ici, c'est simplement un indicateur, qu'il faut penser à multiplier par le nombre de nuits ... (donc pour l'id_tarif courant). A la fin de la facture globale du client, celui ci pourra constater qu'il y a un certain nombre de nuits en saison. Justifiant ainsi le prix global.
- R9 : maintenant que l'on a tout ce qu'il faut, on insère le résultat dans la table ``Lignefacture`` pour `nbnuitt1` ;
- R10 : maintenant que l'on a tout ce qu'il faut, on insère le résultat dans la table ``Lignefacture`` pour `nbnuitt2` ;

3.1.8. TP5 : Choix de la chambre et facturation



Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TP, la suite (examen final + note de TP) dépend de votre compréhension globale.



Et les autres TP ...

C'est une démarche de projet, dans ce cadre les TP sont tous liés les uns aux autres. C'est pour cela, qu'il vous faut avancer correctement, pour ne pas être à un certain moment distancé.

3.1.8.1. Choix de la chambre

Maintenant, la réservation doit fonctionner dans les grandes lignes. Mais comment faire pour choisir la chambre du client ? La réponse est double, en automatique via une requête SQL assez complexe, et en manuelle. En manuelle aussi ? Oui car si le client n'est pas complètement satisfait du mode automatique, il faut qu'il puisse le faire à la main en recherchant toutes les chambres correspondant à ses critères. L'administrateur pourra aussi modifier via l'interface l'attribution de la chambre en question.

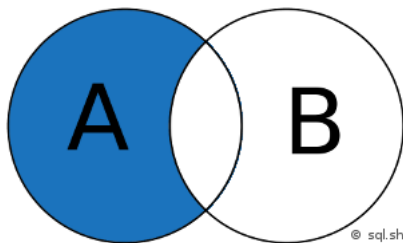
Ceci signifie clairement que vous devez réaliser une interface web permettant de modifier l'attribution de la chambre pour l'administrateur et le client.

Ici, nous allons réfléchir ensemble à la partie attribution automatique. Il faut de suite penser au nombre de lits, car suivant ce critère la chambre n'est pas la même !

```
SELON resa_std_sup_exc
  CAS : 0
    nb_lits = nb_adulte;
  CAS : 1
    nb_lits = nb_adulte - 1;
  CAS : 2
    nb_lits = nb_adulte - 2;
FIN SELON
```

Puis le type de jointure. Ici il s'agit d'une LEFT JOIN !

Figure 3.21. Jointure à gauche



Soit la requête suivante :

```
NEW.id_chambre = (SELECT MIN(C1.id_chambre)
  FROM `Chambre` C1 LEFT JOIN `Reservation` R1 ON C1.`id_chambre` = R1.`id_chambre`
```



```

WHERE C1.`std_sup_exc` = NEW.resa_std_sup_exc AND C1.`nb_lits` >= @nb_lits AND
C1.`dispo` = 1 AND
    (CASE WHEN (C1.`id_chambre` = R1.`id_chambre`) THEN
        (CASE
            WHEN (R1.`date_fin` < NEW.date_debut) THEN true
            WHEN (R1.`date_debut` > NEW.date_fin) THEN true
            ELSE false
        )
    )
ELSE true
END)
);

```

Où @nb_lits est le nombre de lit calculé plus haut, NEW.resa_std_sup_exc est la valeur de resa_std_sup_exc, ou NEW.date_debut est la date_debut et enfin NEW.date_fin est date_fin. Le mot clé MIN permet de récupérer le numéro de chambre le plus bas. Qui est affecté à NEW.id_chambre qui est la réponse à la question.

Pour bien comprendre cette requête, je vous conseille de la décortiquer, en enlevant la fonction MIN, puis en changeant quelque chose afin de vérifier vos hypothèses.



Pour quelles raisons des noms de variables aussi bizarre ?

Bizarre et de plus non fonctionnelle dans votre contexte PHP. C'est parce que j'ai créé de mon côté pour tester un *trigger* en insertion afin que MySQL calcule tout seul le numéro de chambre, cela m'a évité d'écrire du code PHP. C'est de fait mieux, car on utilise toute la puissance du moteur de base de données. Or, les *trigger* ne sont pas au programme de votre formation ...

Vous devez pouvoir répondre aux questions suivantes :

- Qui est à gauche dans la requête, qui est A et qui est B ?
- A quoi servent les parties WHEN (R1.`date_fin` < NEW.date_debut) THEN true du CASE ?
- Vous devez avoir tout compris de cette requête, vous avez un examen final me semble t-il ?

Après vous intégrerez cette requête dans votre classe 'Reservation' qui sera appelée lors de la construction de l'objet. Donc, vous devez modifier le code du TP précédent.

3.1.8.2. La facturation

Vous devez fournir au client la facture de sa réservation en ligne. L'idéal serait de la lui donner sous forme de fichier PDF. Si vous avez le temps ... Faites le !

Méthode de travail :

- Vous devez dans un premier temps traiter toutes les lignes de la table 'Lignefacture' concernant le client en question. Ceci signifie, que vous devez faire les totaux de ses périodes de réservation. Le moteur de base de données c'est faire des calculs, utiliser donc cet aspect ;
- Vous devez réaliser l'interface web permettant de visualiser toutes les informations le concernant ;
- Pensez que les lignes de la table 'Lignefacture' doivent apparaître dans la facture finale ;
- Pensez que l'administrateur doit pouvoir lui attribuer une remise ;
- Vous devez coder une classe 'Facturation' contenant toutes les méthodes permettant de gérer la facturation ;
- Un fichier PDF de la facture ? Vous pouvez si vous le souhaitez ! Si vous allez assez loin dans ce travail, vous pourrez très certainement ensuite proposer des facturations en lignes à vos clients si vous décidez d'ouvrir votre boîte.

3.1.9. TP6 : Synthèse

3.1.9.1. La mise au point de la BDD

Le moteur de la BDD est InnoDB, vous avez donc la possibilité de gérer les clés étrangères, et c'est bien évidemment le but de ce moteur. Pouvoir faire faire des tâches en automatique.

Règles liées au comportement de la BDD :

- Si l'on supprime un client de la table 'Client', toutes les informations relatives à son passage dans l'hôtel doivent être supprimées (réservations, factures, et son compte). Soient les tables 'Lignefacture', 'Facture' et 'Compte' ;
- Si l'on supprime une réservation celle-ci disparaît de la table 'Lignefacture' et de la table 'Facture' ;
- On ne peut pas supprimer une facture de la table 'Facture' directement, il faut d'abord supprimer les lignes de la tables 'Lignefacture' concernées par celle ci ;
- On ne peut pas supprimer un 'Compte' directement car il est lié à la table 'Client' ;

3.1.9.2. La partie administration du site

Maintenant, vous devez proposer une interface web pour l'administrateur avec un menu sans fioritures. Un menu professionnel, cette partie ne doit pas ressembler au site de l'hôtel. Pas d'image en haut, quelque chose de sobre.



Vous avez dit professionnel ...

L'administrateur du site de réservation de chambre d'hôtel n'est pas informaticien, il doit donc se retrouver sur une interface d'administration simple claire et efficace.

Vous avez sur le site de Foundation du code intéressant pour réaliser un Dashboard [<https://foundation.zurb.com/building-blocks/kits/dashboard.html>] vous gagnerez à vous en inspirer.

Qu'est ce que souhaiterait avoir comme interface un administrateur de site de réservation en ligne ?

- Avoir un accès complet à toutes les tables afin de pouvoir modifier leur contenu directement :
 - Pour la table 'Compte' pouvoir changer login / password, et donc de fait son propre compte ;
 - Pour la table 'Client' pouvoir changer l'un des champs, et supprimer un compte 'Client' impactant donc automatiquement la table 'Compte' ;
 - Pouvoir accorder une remise à un client (table 'Facture'), valider le paiement (modification du champ paiement_ok) ;
 - Pouvoir supprimer une réservation (table 'Reservation') ou la modifier impactant alors les tables 'Lignefacture' et 'Facture' ;
 - Pouvoir ajouter et supprimer des champs de la table 'Tarif' ;
- Connaître en continue le nombre de réservations de la journée avec la précision a réglé ou non le séjour ;
- Avoir une courbe permettant de connaître en temps réel l'occupation de l'hôtel, pour cela vous pouvez utiliser JpGraph [https://www.phpfacile.com/apprendre_le_php/courbes_et_histogrammes_avec_jpgraph] ...

Références

[1] . Les jointures [<https://sql.sh/cours/jointures>] .

- [2] . pdo-master [<https://github.com/taniascia/pdo/tree/feba19a43e63c617c95a1f6e68825f6e3086336c>] .
- [3] . Tutoriel pdo-master [<https://www.taniascia.com/create-a-simple-database-app-connecting-to-mysql-with-php/>] .
;
- [4] . ZURB Foundation [<https://foundation.zurb.com/>] .
- [5] . XY Grid [<https://foundation.zurb.com/sites/docs/xy-grid.html>] .
- [6] . <https://www.smarty.net/docsv2/fr/> [<https://www.smarty.net/docsv2/fr/>] .
- [7] . Web 2.0, allez plus loin avec AJAX et XMLHttpRequest [<https://siddh.developpez.com/articles/ajax/>] .
- [8] . Site officiel du *Le Grand Hôtel* [<http://www.grand-hotel-perros-guirec.com>] .
- [9] . Une machine virtuelle bien utile [<http://partage.rt-iut.re/rtcloud/Machines-Virtuelles/VM-Linux/devuan/>] .
- [10] . Lien pour télécharger AnalyseSI [<https://launchpad.net/analysesi/+download>] .
- [11] . Codes sources et début d'analyse MCD (v4) [<https://drive.google.com/drive/folders/1B3o3UTt7Btuf08B4mGrZyo-C6MiiNCYd?usp=sharing>] .
- [12] . Reveal | Foundation for Sites [<https://foundation.zurb.com/sites/docs/reveal.html>] .
- [13] . Dropdown [<https://foundation.zurb.com/sites/docs/dropdown.html>] .
par exemple !
- [14] . Abide démo [https://www.tutorialspoint.com/foundation/abide_demo.htm] .
- [15] . Un exemple avec Foundation 5.5.3 (identique à la dernière version) [<https://foundation.zurb.com/sites/docs/v/5.5.3/components/abide.html>] .
- [16] . Doc officielle Foundation zurb [<https://foundation.zurb.com/sites/docs/abide.html>] .
- [17] . Dashboard [<https://foundation.zurb.com/building-blocks/kits/dashboard.html>] .
- [18] . JpGraph [https://www.phpfacile.com/apprendre_le_php/courbes_et_histogrammes_avec_jpgraph] .
- [19] . *Dichotomie* - Wikipédia, l'encyclopédie libre et gratuite [<http://http://fr.wikipedia.org/wiki/Dichotomie>] .
- [20] . Une machine virtuelle bien utile [<http://partage.rt-iut.re/rtcloud/Machines-Virtuelles/VM-Linux/devuan/>] .
- [21] . Lien pour télécharger AnalyseSI [<https://launchpad.net/analysesi/+download>] .
- [22] . Codes sources et début d'analyse MCD (v4) [<https://drive.google.com/drive/folders/1B3o3UTt7Btuf08B4mGrZyo-C6MiiNCYd?usp=sharing>] .
- [23] . Site officiel du *Le Grand Hôtel* [<http://www.grand-hotel-perros-guirec.com>] .
- [24] . TripAdvisor *Le Grand Hôtel* [https://www.tripadvisor.fr/Hotel_Review-g207340-d1024359-Reviews-Le_Grand_Hotel-Perros_Guirec_Cotes_d_Armor_Brittany.html] .
- [25] . LogiTravel *Le Grand Hôtel* [<https://www.logitravel.fr/hotels/europe/france/perros-guirec/le-grand-hotel-trestraou-17862907.html>] .
- [26] . Expédia.fr *Le Grand Hôtel* [<https://www.expedia.fr/Cotes-DArmor-Hotel-Grand-Hotel-De-Trestraou.h23478575.Description-Hotel>] .
- [27] . Foundation - Top Bar with Responsive Toggle and Animation (CSS/Float Example) [<https://codepen.io/IamManchanda/pen/KmepBg>] .
- [28] . Foundation 6 Megamenu Test [<https://codepen.io/madsciencepro/pen/WRwyvP>] .
- [29] . Foundation 6 Menu Examples [<https://codepen.io/SitePoint/full/bEYPmg/>] .

[30] . Foundation 6 Top Bar [codepen.io/shoaibik/pen/wGORZY] .