

TUMBARELLO Mathéo  
CHASSANY Jules  
PHONGSAVANG Marianna  
HAQUIN Syrine  
Groupe 22



SAE 2.03 installation de services réseau

Année Universitaire 2021-2022

---

# SAE 2.03 - COMPTE-RENDU PROJET

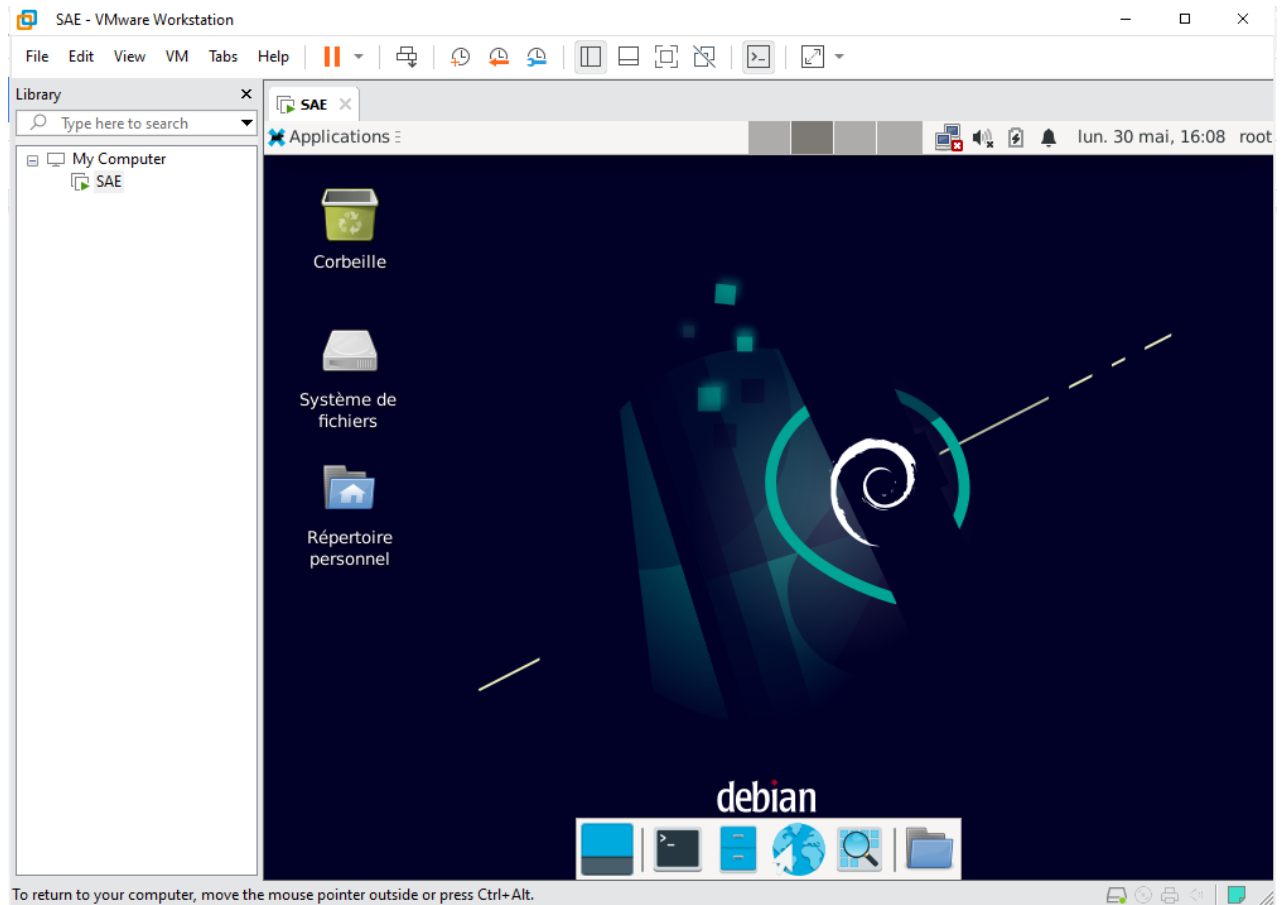
## **SOMMAIRE**

<b>TP</b>	<b>2</b>
<b>Tâches et répartition réelle</b>	<b>11</b>
<b>Gestion de projet</b>	<b>12</b>
Idée de site ?	12
Méthode de travail retenue	12
Gantt prévisionnel	12
Gantt final	13
<b>Fiche de suivi</b>	<b>14</b>
Rôles	14
Suivi	14
<b>Configuration Apache nécessaires</b>	<b>17</b>
Système de connexion à une page d'administrateur	17
Directories sans index	18
Personnaliser les pages d'erreurs	19
<b>Base de données</b>	<b>20</b>
Conception	20
Implémentation	21
<b>PHP</b>	<b>22</b>
Glossaire des fonctions	22
Connexion à la base (PDO)	24
Page d'accueil	25
Inscription	26
Lien de confirmation	27
Connexion de l'utilisateur	28
Déconnexion de l'utilisateur	29
Sessions et redirections automatiques	30
Gérer les annonces	30
Créer une annonce	30
Recherche les annonces avec une catégorie	32
Supprimer une annonce	33
Page administrateur	34
Gérer les catégories	35
Créer une catégorie	35
Supprimer une catégorie	35
<b>CRON</b>	<b>38</b>
<b>Approfondissements possibles</b>	<b>39</b>
<b>BILAN</b>	<b>40</b>
<b>Bibliographie</b>	<b>41</b>

# TP

Username: root

Password: lannion



2.

Activité depuis 7 minutes.

Heure de lancer: 16h05min24sec

```

root@SAE:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pr
   Active: active (running) since Mon 2022-05-30 16:05:24 CEST; 7min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 534 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/
   Process: 949 ExecReload=/usr/sbin/apachectl graceful (code=exited, statu
   Main PID: 703 (apache2)
      Tasks: 6 (limit: 2285)
     Memory: 18.6M
        CPU: 204ms
    CGroup: /system.slice/apache2.service
            └─703 /usr/sbin/apache2 -k start
              └─968 /usr/sbin/apache2 -k start
                └─969 /usr/sbin/apache2 -k start
                  └─970 /usr/sbin/apache2 -k start
                    └─971 /usr/sbin/apache2 -k start
                      └─972 /usr/sbin/apache2 -k start

mai 30 16:05:23 SAE systemd[1]: Starting The Apache HTTP Server...
mai 30 16:05:24 SAE apachectl[567]: AH00558: apache2: Could not reliably det
mai 30 16:05:24 SAE systemd[1]: Started The Apache HTTP Server.
mai 30 16:05:28 SAE systemd[1]: Reloading The Apache HTTP Server.
mai 30 16:05:28 SAE apachectl[959]: AH00558: apache2: Could not reliably det
lines 1-23...skipping...

```

date de

lancer: 30/05/2022

2. Serveur actif depuis le 30/05/2022 à 16:05:24h

3.

- cd /etc/apache2/sites-available/
- cat 000-default.conf (affiche où se trouve le document root -> '/var/www/html')
- cd /var/www/html
- cat index.html

Le DocumentRoot sous Apache2 permet de connaître la racine de ce qui est accessible depuis le localhost.

4. type -a apache2

```

root@SAE:/sbin# type -a apache2
apache2 est /usr/sbin/apache2
apache2 est /sbin/apache2

```

5.

```

root@SAE:/var/www/html# apache2 -v
Server version: Apache/2.4.53 (Debian)
Server built: 2022-03-14T16:28:35

```

6.

```

root@SAE:/var/www/html# apache2 -l
Compiled in modules:
  core.c
  mod_so.c
  mod_watchdog.c
  http_core.c
  mod_log_config.c
  mod_logio.c
  mod_version.c
  mod_unixd.c

```

7. Il sert à lister tous les retours (logs) de la configuration durant l'utilisation du serveur apache2.

8.

```

root@SAE:/sbin# apachectl -M
AH00558: apache2: Could not reliably
globally to suppress this message
Loaded Modules:
  core_module (static)
  so_module (static)
  watchdog_module (static)
  http_module (static)
  log_config_module (static)
  logio_module (static)
  version_module (static)
  unixd_module (static)
  access_compat_module (shared)
  alias_module (shared)
  auth_basic_module (shared)
  authn_core_module (shared)
  authn_file_module (shared)
  authz_core_module (shared)
  authz_host_module (shared)
  authz_user_module (shared)
  autoindex_module (shared)
  deflate_module (shared)
  dir_module (shared)
  env_module (shared)
  filter_module (shared)
  mime_module (shared)
  mpm_prefork_module (shared)
  negotiation_module (shared)
  php7_module (shared)
  reqtimeout_module (shared)
  setenvif_module (shared)
  status_module (shared)

```

9.

```
root@SAE:/sbin# apache2 -v
Server version: Apache/2.4.53 (Debian)
Server built: 2022-03-14T16:28:35
root@SAE:/sbin# apache2 -version
Server version: Apache/2.4.53 (Debian)
Server built: 2022-03-14T16:28:35
```

a)

Création partie admin:

→ Ajout d'un dossier *private* au niveau du DocumentRoot.

→ Créer un user avec htpass (htpasswd -c /etc/apache2/pass <user>

→ Aller dans /etc/apache2/apache2.conf

→ Aller à l'endroit où les directories sont renseignés et y ajouter le nouveau directory créé en ligne de commande en faisant en sorte de l'imposer en privé.

(Authentification gérée par Apache2)

→ taper localhost dans la barre de recherche (il est parfois nécessaire de vider le cache.) et se connecter lorsqu'on tente d'accéder à une page du dossier *private*.

## Module PHP:

1. N/A

2. Il est recommandé de créer un dossier secret car il assure une sécurité supplémentaire face aux tentatives d'accès à des fichiers/pages. Si une quelconque personne affiche l'ensemble de la hiérarchie complète du site, il ne pourra normalement pas voir les dossiers secrets. Cependant, il pourra toujours y accéder via l'url, c'est pourquoi il serait plus judicieux de donner des noms compliqués à ce type de dossiers.

3. localhost/.php/phpinfo.php

4. On voit une page d'info php. Techniquement, si on réussit à y accéder via l'url, cela veut dire qu'il est activé puisque cette page provient d'une fonction php exécutée par le serveur lorsque la page est actualisée.

5.

→ modules activées : /etc/apache2/mods-enabled

→ vérifier fichier php.7.4.load

→ Ce fichier est un raccourci vers le fichier '../mods-available/php7.4.load' (celui-ci charge le module PHP.)

6. Accédez à votre script php : phpinfo.php avec le navigateur et relevez les informations suivantes sur la page affichée par le serveur :

- i. La version exacte du module PHP utilisé par notre serveur Web  
→ PHP Version 7.4.28
- ii. Le dossier de configuration du module PHP utilisé par notre serveur Web  
→ /etc/php/7.4/apache2
- iii. Le fichier de configuration de php pour le serveur Web  
→ /etc/php/7.4/apache2/php.ini
- b. La valeur de l'étiquette appelée « short\_open\_tag »  
→ Off
- c. À votre avis, dans quel fichier de configuration (chemin exacte) peut-on modifier la valeur de cette étiquette « short\_open\_tag » ?  
→ /etc/php/7.4/apache2/php.ini, oui, nous pouvons modifier la valeur du short\_open\_tag.
- d. Si on devait modifier la valeur de cette étiquette dans un fichier de configuration, faut-il recharger/relancer le serveur Web pour que la nouvelle valeur soit prise en compte ?  
→ oui il le faut toujours grâce à la commande # systemctl restart apache2

7. chemin du binaire : /usr/bin/php7.4

dans le dossier /usr/bin/ : php7.4 -v → version : 7.4.28

8.

```
root@SAE:/usr/bin# a2dismod php7.4
```

Module php7.4 disabled.

To activate the new configuration, you need to run:

```
systemctl restart apache2
```

9.

Oui, il faut relancer le serveur Apache puisqu'on vient modifier l'état d'un module dans les configurations d'Apache.

10.

Conséquences de la désactivation du module PHP : On ne peut plus accéder aux pages php depuis l'URL puisque l'interpréteur est désactivé. C'est évidemment dangereux puisque aucun script ne pourra être exécuté sur le serveur (côté back) et créer des problèmes de compatibilité.

- accès URL : impossible
- fichiers apache : aucun fichier de php n'est dans le dossier /etc/apache2/mods\_enabled.

11.

- a. Oui il faut encore redémarrer le serveur
- b. - accès URL : possible  
- fichiers apache : fichier php dans le dossier /etc/apache2/mods\_enabled

12.

*"hp echo "coucou, je suis un code  
php dans une page HTML !" ?>"*

13.

Oui car on vient de modifier une configuration.

14.

Cela ne fonctionne pas car le serveur web ne reconnaît pas le php à travers le .html

15. La page mapage.html Fonctionne correctement et affiche les message et programme php.

Dans etc/apache2/mods\_available/php7.4.conf

```
<FilesMatch ".html">
```

```
    SetHandler application/x-httpd-php
```

```
</FilesMatch>
```

16.

Le fichier n'est pas reconnu car nous n'avons pas indiqué au module php qu'il pouvait lire les fichiers sans extension.

17. test confirm: page phpinfo comprise par le navigateur

```
<FilesMatch "$">
```

```
    SetHandler application/x-httpd-php
```

```
</FilesMatch>
```

18.

- a) *<? echo "coucou, je suis un code php dans une page HTML!"?>* : intégré à la page mapage.html
- b) mapage.html n'est pas lu par le navigateur car il ne reconnaît aucun langage
- c) modifier /etc/php/7.4/apache2/php.ini et trouver short\_open\_tag, le changer en short\_open\_tag=ON

## MySQL & PHP :

1.

#mysql\_secure\_installation



2.

```
root@SAE:~# systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2022-06-07 08:33:41 CEST; 14min ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
   Process: 538 ExecStartPre=/usr/share/mysql-8.0/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
  Main PID: 654 (mysqld)
    Status: "Server is operational"
     Tasks: 37 (limit: 2285)
    Memory: 446.6M
       CPU: 3.190s
   CGroup: /system.slice/mysql.service
           └─654 /usr/sbin/mysqld

Jun 07 08:33:35 SAE systemd[1]: Starting MySQL Community Server...
Jun 07 08:33:41 SAE systemd[1]: Started MySQL Community Server.
```

systemctl status mysql

→ Le service MySQL se lance dès qu'on lance la VM. En effet, le serveur est en cours d'exécution dès le lancement de celle-ci, MySQL étant un service lié au serveur, il prend cet état.

3.

```
root@SAE:~# mysql -V
mysql Ver 8.0.29 for Linux on x86_64 (MySQL Community Server - GPL)
```

4.

Connexion à MySQL : `mysql -u <user> -p`

5.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0,00 sec)
```

6.

a. Dans une page web, intégrer ce code :

```
<?php
    $sql="SHOW DATABASES";
    $link = mysqli_connect('10.1.2.3', 'root',
        'password') or die ('Error connecting to mysql: ' .
        mysqli_error($link) . '\r\n');
    if (!($result=mysqli_query($link,$sql))) {
```

```

        printf("Error: %s\n", mysqli_error($link));
    }
    while( $row = mysqli_fetch_row( $result ) ){
        if ( ($row[0]!="information_schema") &&
            ($row[0]!="mysql")) {
            echo $row[0]."\r\n";
        }
    }
}
?>

```

b. Les packages php-mysql ne sont pas installés et ne sont donc pas supportés.

c.

```

dpkg -i /usr/local/src/php7.4-mysql_7.4.28-1+deb11u1_amd64.deb
dpkg -i /usr/local/src/php-mysql_7.4+76_all.deb
systemctl restart apache2

```

## 7. Créer une nouvelle DataBase:

```
Create Database <namedtb>;
```

Créer un nouveau schéma:

```
Create Schema <prenom>;
```

## 8. Créer une nouvelle Table:

```

create table matheo.etudiants(id INT PRIMARY KEY NOT NULL,nom
VARCHAR(100),date_naissance VARCHAR(10), classement
VARCHAR(255)) ;

```

## Insérer Valeurs:

```

INSERT INTO matheo.etudiants(id, nom,date_naissance,
classement)
VALUES
('1', 'Armand', '25/12/1993',53),
('2', 'Hebert', '12/03/2000', 36),
('3', 'Savary', '14/08/1983', 12);

```

```
select * from matheo.etudiants;
```

## 9. Supprimer values of table:

```

delete from matheo.etudiants where id = 1;
select count(*) from matheo.etudiants; (pour vérifier qu'une ligne a été
supprimée.)

```

10.

```
UPDATE matheo.etudiants set date_naissance = '01/01/1990'
where id in (select min(id) from (select id from
matheo.etudiants) as e);
```

11.

Insérer des valeur a travers un script php:

```
$sql="INSERT INTO matheo.etudiants(id, nom,date_naissance,
classement)VALUES('4', 'Armand', '25/12/1993',53);";
if (!($result=mysqli_query($link,$sql))) {
    printf("Error: %s\n", mysqli_error($link)); }
```

Modifier des valeurs a travers un script php:

```
$sql="UPDATE matheo.etudiants set date_naissance =
'25/10/2003' where id in (select min(id) from (select id from
matheo.etudiants) as e);";
if (!($result=mysqli_query($link,$sql))) {
    printf("Error: %s\n", mysqli_error($link)); }
```

Observer les valeurs à travers un script PHP:

```
$sql="select * from gontrand.etudiants;";
$resultat = mysqli_query($link, $sql);
while($row = mysqli_fetch_array($resultat))
{
    echo $row['id'] ." ". $row['nom'] ." ".
$row['date_naissance'] ." ". $row['classement'] . "<br>";
}
```

# Tâches et répartition réelle

## Front-end

- Page d'accueil (Syrine, Marianna)
- Page d'inscription (Syrine)
- Page de login (Marianna)
- Page d'administrateur (Syrine, Marianna)
- Page d'erreur 404 (Marianna)
- Page d'ajout d'une annonce (Syrine)
- Page de l'utilisateur (avec la liste de ses annonces déjà publiées) (N/A)
- Contenu éditorial (Syrine, Marianna)

## Back-end

- Diagramme UML BDD (Tous)
- Construction de la BDD (Syrine, Marianna)
- Script de connexion à la BDD (Mathéo, Jules)
- Programmation du lien de confirmation + validation utilisateur (Mathéo, Jules)
- Programmation inscription (Mathéo, Jules)
- Programmation connexion utilisateur (Mathéo, Jules)
- Programmation accès pages avec authentification nécessaire et les redirections en cas d'utilisateurs anonymes tentant d'y accéder (Mathéo)
- Programmation de la suppression des utilisateurs chaque jour à 3h du matin qui se sont inscrits dont le statut n'est pas "validé". (Jules)
- Programmation du système d'ajout d'annonce et d'enregistrement dans la base (Jules)
- Programmation du système de gestion de catégorie (Mathéo)

## Gestion de projet

- Gantt prévisionnel (Marianna)
- Gantt final (Marianna)
- Rédaction du compte-rendu (Tous)
- Mettre à jour continuellement la fiche de suivi (Syrine)

# Gestion de projet

## Idée de site ?

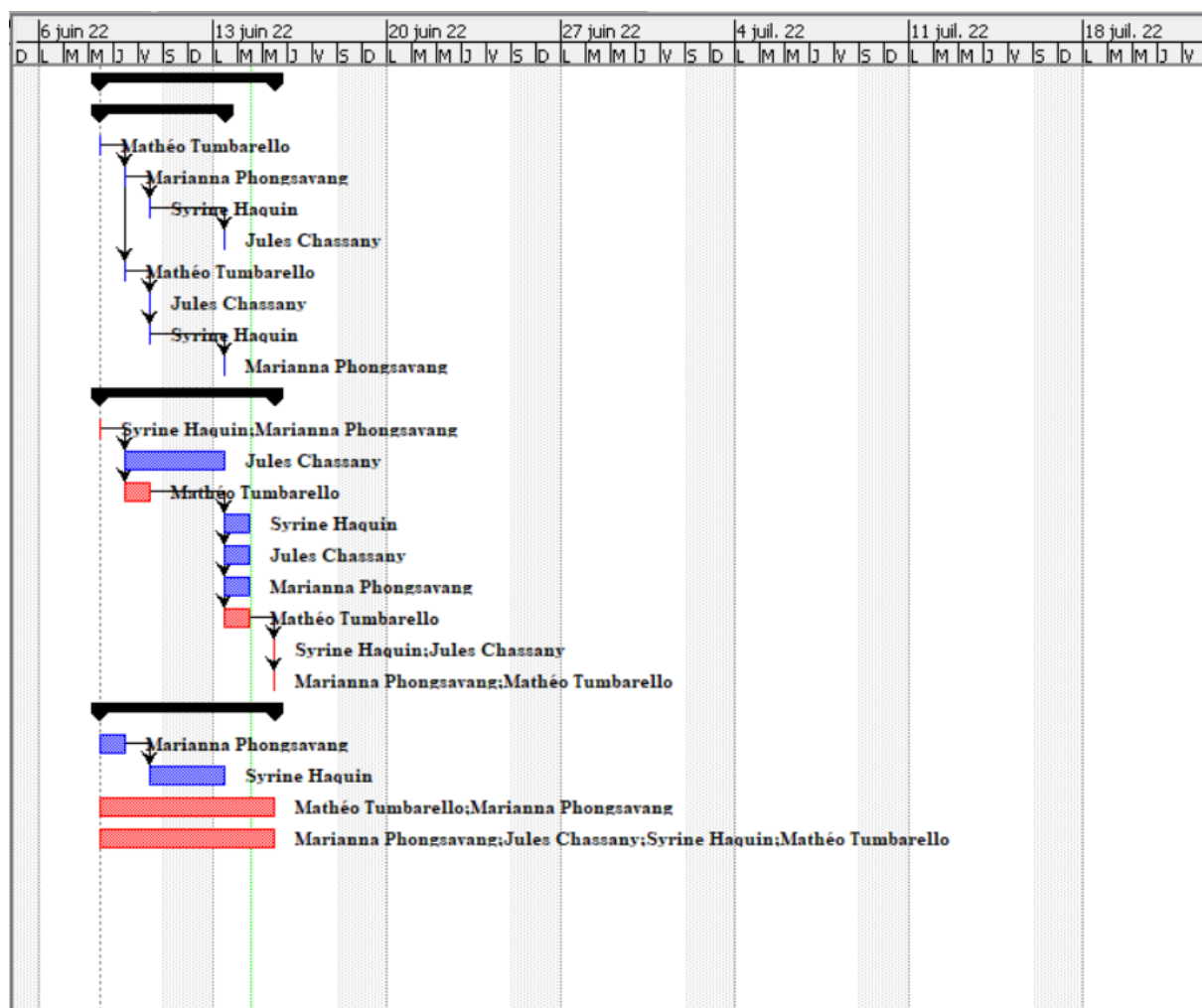
Afin de répondre aux besoins du cahier des charges, nous avons réfléchi sur le développement d'un site dynamique pouvant susciter de l'intérêt chez n'importe quel utilisateur. C'est pourquoi nous nous sommes penchés vers la réalisation d'un site de poste d'annonces de tout type répertoriées dans des catégories uniques.

## Méthode de travail retenue

Après la réalisation du TP, nous nous sommes mis d'accord sur la façon dont nous allons travailler pour être efficaces en vue du délai très court imposé. Ainsi, nous avons très souvent organisé des binômes de deux, chacun des binômes s'occupant soit séparément de tâches simples de même type, soit ensemble dans le cas de tâches plus complexes afin de permettre de réfléchir à plusieurs, d'optimiser les idées données, et de toujours avoir un avis extérieur.

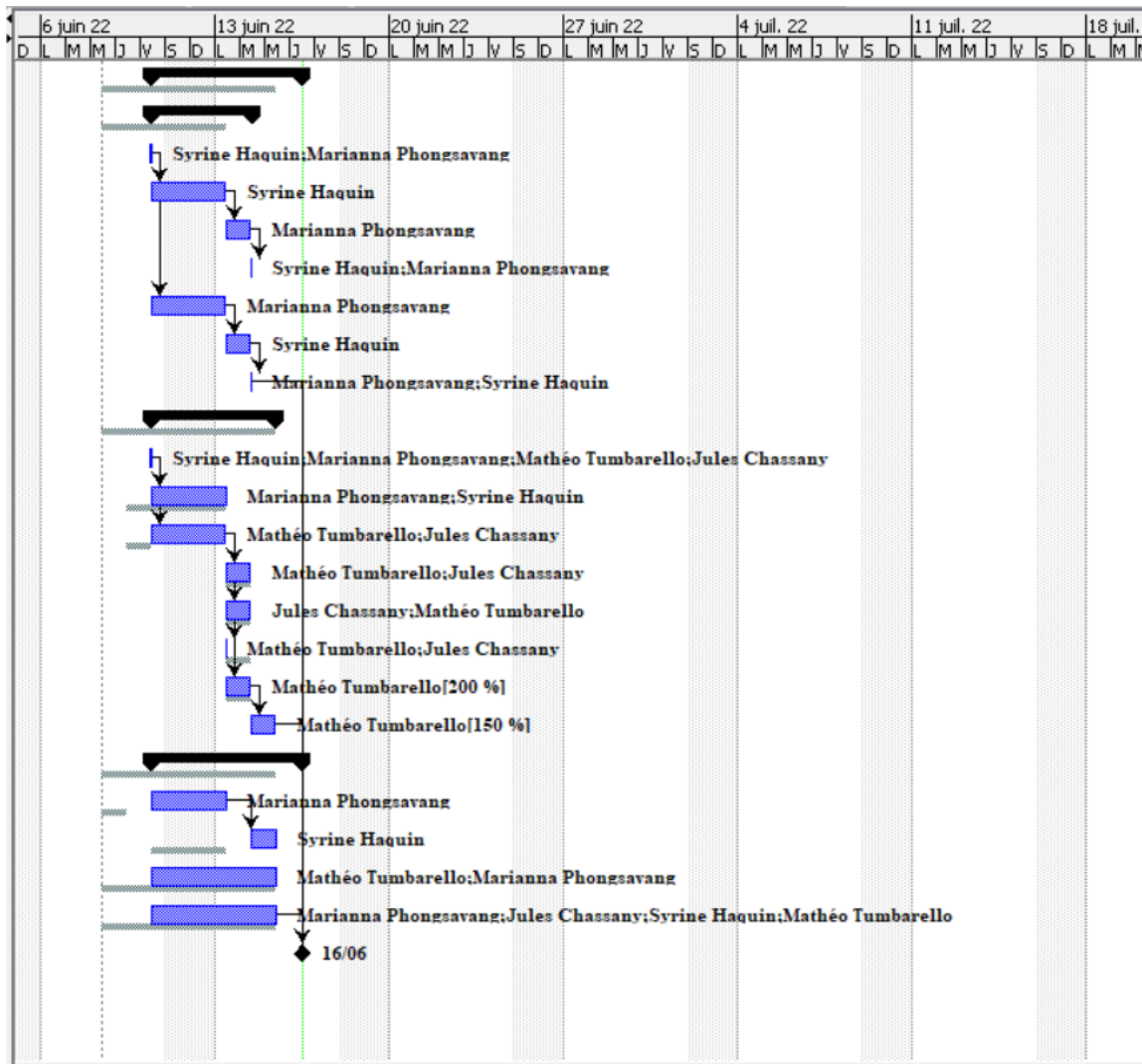
## Gantt prévisionnel

	🕒	Nom	Durée	Début	Fin	Prédécesseurs	Noms des ressources
1		📁 Réalisation du site	6 jours	08/06/22 11:00	15/06/22 12:00		
2		📁 Front-end	4 jours	08/06/22 11:00	13/06/22 12:00		
3		Page d'accueil	1 jour	08/06/22 11:00	08/06/22 12:00		Mathéo Tumbarello
4		Page d'inscription	1 jour	09/06/22 11:00	09/06/22 12:00	3	Marianna Phongsavang
5		page de login	1 jour	10/06/22 11:00	10/06/22 12:00	4	Syrine Haquin
6		page administrateur	1 jour	13/06/22 11:00	13/06/22 12:00	5	Jules Chassany
7		page d'erreur	1 jour	09/06/22 11:00	09/06/22 12:00	3	Mathéo Tumbarello
8		page d'ajout d'une annon	1 jour	10/06/22 11:00	10/06/22 12:00	7	Jules Chassany
9		page de l'utilisateur	1 jour	10/06/22 11:00	10/06/22 12:00	7	Syrine Haquin
10		contenu editorial	1 jour	13/06/22 11:00	13/06/22 12:00	9	Marianna Phongsavang
11		📁 Back-end	6 jours	08/06/22 11:00	15/06/22 12:00		
12		Diagramme UML BDD	1 jour	08/06/22 11:00	08/06/22 12:00		Syrine Haquin;Marianna Pho...
13		Construction de la BDD	3 jours	09/06/22 11:00	13/06/22 12:00	12	Jules Chassany
14		Script de connexion à la B	2 jours	09/06/22 11:00	10/06/22 12:00	12	Mathéo Tumbarello
15		Programmation du lien pe	2 jours	13/06/22 11:00	14/06/22 12:00	14	Syrine Haquin
16		Programmation inscription	2 jours	13/06/22 11:00	14/06/22 12:00	14	Jules Chassany
17		Programmation connexion	2 jours	13/06/22 11:00	14/06/22 12:00	14	Marianna Phongsavang
18		Programmation authentifi	2 jours	13/06/22 11:00	14/06/22 12:00	14	Mathéo Tumbarello
19		Programmation suppression	1 jour	15/06/22 11:00	15/06/22 12:00	18	Syrine Haquin;Jules Chassany
20		Programmation du system	1 jour	15/06/22 11:00	15/06/22 12:00	18	Marianna Phongsavang;Mat...
21		📁 Gestion de projet	6 jours	08/06/22 11:00	15/06/22 12:00		
22		Gantt previsionnel	2 jours	08/06/22 11:00	09/06/22 12:00		Marianna Phongsavang
23		Gantt finale	2 jours	10/06/22 11:00	13/06/22 12:00	22	Syrine Haquin
24		Redaction du compte rend	6 jours	08/06/22 11:00	15/06/22 12:00		Mathéo Tumbarello;Mariann...
25	📅	Fiche de suivi	6 jours	08/06/22 11:00	15/06/22 12:00		Marianna Phongsavang;Jule...



## Gantt final

	①	Nom	Durée	Début	Fin	Prédécesseurs	Noms des ressources
1	📅	☐ Réalisation du site	5 jours	10/06/22 11:00	16/06/22 12:00		
2		☐ Front-end	3 jours	10/06/22 11:00	14/06/22 12:00		
3		Page d'accueil	0,5 jours	10/06/22 11:00	10/06/22 11:30		Syrine Haquin; Marianna Pho...
4		Page d'inscription	1 jour	10/06/22 11:30	13/06/22 11:30	3	Syrine Haquin
5		page de login	1 jour	13/06/22 11:30	14/06/22 11:30	4	Marianna Phongsavang
6		page administrateur	0,5 jours	14/06/22 11:30	14/06/22 12:00	5	Syrine Haquin; Marianna Pho...
7		page d'erreur	1 jour	10/06/22 11:30	13/06/22 11:30	3	Marianna Phongsavang
8		page d'ajout d'une annon	1 jour	13/06/22 11:30	14/06/22 11:30	7	Syrine Haquin
9		contenu editorial	0,5 jours	14/06/22 11:30	14/06/22 12:00	8	Marianna Phongsavang; Syri...
10		☐ Back-end	3,833 jours	10/06/22 11:00	15/06/22 11:50		
11		Diagramme UML BDD	0,5 jours	10/06/22 11:00	10/06/22 11:30		Syrine Haquin; Marianna Pho...
12		Construction de la BDD	1,5 jours	10/06/22 11:30	13/06/22 12:00	11	Marianna Phongsavang; Syri...
13		Script de connexion à la b	1 jour	10/06/22 11:30	13/06/22 11:30	11	Mathéo Tumbarello; Jules Ch...
14		Programmation du lien pe	1 jour	13/06/22 11:30	14/06/22 11:30	13	Mathéo Tumbarello; Jules Ch...
15		Programmation inscription	1,333 jours	13/06/22 11:30	14/06/22 11:50	13	Jules Chassany; Mathéo Tu...
16		Programmation connexion	0,5 jours	13/06/22 11:30	13/06/22 12:00	13	Mathéo Tumbarello; Jules Ch...
17		Programmation authentif	1 jour	13/06/22 11:30	14/06/22 11:30	13	Mathéo Tumbarello[200 %]
18		Programmation du systè	1,333 jours	14/06/22 11:30	15/06/22 11:50	17	Mathéo Tumbarello[150 %]
19		☐ Gestion de projet	5 jours	10/06/22 11:00	16/06/22 12:00		
20		Gantt previsionnel	2 jours	10/06/22 11:00	13/06/22 12:00		Marianna Phongsavang
21		Gantt finale	2 jours	14/06/22 11:00	15/06/22 12:00	20	Syrine Haquin
22		Redaction du compte ren	4 jours	10/06/22 11:00	15/06/22 12:00		Mathéo Tumbarello; Mariann...
23	📅	Fiche de suivi	4 jours	10/06/22 11:00	15/06/22 12:00		Marianna Phongsavang; Jule...
24	📅	soutenance	0 jours	16/06/22 12:00	16/06/22 12:00	23; 18; 9	



## Fiche de suivi

### Rôles

- Tumbarello Mathéo : chef de projet
- Phongsavang Marianna : chef-adjoint
- Haquin Syrine : secrétaire
- Chassany Jules : développeur

### Suivi

### Composition de votre groupe

Nom et prénom	Numéro affecté	Demi Groupe TP
CHASSANY Jules	1	G2
HAQUIN Syrine	2	G2
TUMBARELLO Mathéo	3	G2
PHONGSAVANG Marianna	4	G2

État de Présence :

#Etudiant	Séance 1	Séance 2	Séance 3	Séance 4	Séance 5
1	Oui	Oui	Oui	Oui	Oui
2	Oui	Oui	Oui	Oui	Oui
3	Oui	Oui	Oui	Oui	Oui
4	Oui	Oui	Oui	Oui	Oui

Résumés des avancées de chaque séanc TD, TP (encadré) ou Projet (encadré)

<p><b>Séance 1 (encadrée) (17/05/2022)</b></p> <p><i>Résumé des tâches effectuées :</i></p> <p>Création des groupes et prise de connaissance du sujet</p>
<p><i>Résumé des tâches identifiées pour la séance suivante :</i></p> <p>Commencement du tp de la sae</p>



## Séance 2 (encadrée) (30/05/2022)

### *Résumé des tâches effectuées :*

tp SAE commencé jusqu'à question 9 de la partie Apache :

- création de la VM
- prise en main de la VM et de Apache

### *Résumé des tâches identifiées pour la séance suivante :*

continuation du tp

## Séance 3 (encadrée) (31/05/2022)

### *Résumé des tâches effectuées :*

Continuation du tp SAE jusqu'à la question 8 de la partie PHP

- activation du PHP sur notre serveur WEB
- comprendre son chemin dans les dossiers et ses configurations

### *Résumé des tâches identifiées pour la séance suivante :*

Continuation du tp SAE

## Séance 4 (encadrée) (07/06/2022)

### *Résumé des tâches effectuées :*

Continuation du tp SAE jusqu'à la question 1 de la partie MySQL et PHP :

- test de désactivation du module PHP
- création d'une nouvelle page HTML avec du PHP
- changement de configuration sur le module PHP pour le short\_open\_tag

### *Résumé des tâches identifiées pour la séance suivante :*

Terminer le TP sae

Se renseigner et se documenter à propos des outils que nous allons utiliser

Trouver une idée pour le site

Réaliser le diagramme de GANTT

Réaliser le site complet

Rédiger le compte-rendu

Réaliser le diaporama

## Configuration Apache nécessaires

### Système de connexion à une page d'administrateur

Pour créer un dossier privé dont la connexion est gérée par Apache, on passe par le terminal et la configuration Apache.

→ Création d'un dossier private dans le projet sae203 (*sae203/private*)

→ Création d'un user via htpass (`htpasswd -c /etc/apache2/pass <user>`)

→ Ouvrir le fichier de configuration Apache (`/etc/apache2/apache2.conf`)

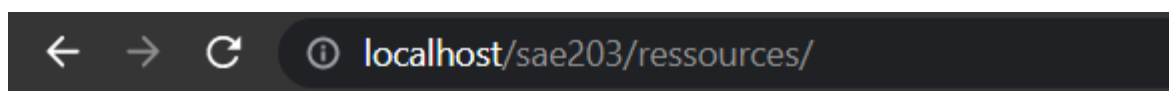
→ Créer un nouveau directory dont l'accès n'est réservé qu'au user qu'on vient de créer :

```
<Directory /var/www/html/sae203/private>
AuthType Basic
AuthName "Administration coin BZH"
AuthUserFile /etc/httpd/htpasswd
Require valid-user
</Directory>
```






→ Vider le cache web puis tapez *localhost* dans l'URL, puis tenter d'accéder à *private*.  
Authentification Apache2 nécessaire pour y accéder.

## Directories sans index

On remarque qu'en essayant de naviguer directement dans un dossier sans index, on accède à la liste complète de tous les dossiers/fichiers répertoriés à l'intérieur de celui-ci, ce qui constitue une faille de sécurité évidente puisque n'importe qui aurait donc accès à la hiérarchie générale des fichiers.



# Index of /sae203/ressources

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">fic.php</a>	2022-06-15 18:13	64	
 <a href="#">img/</a>	2022-06-15 18:12	-	
 <a href="#">includes/</a>	2022-06-15 18:12	-	
 <a href="#">libs/</a>	2022-06-15 18:11	-	

*Apache/2.4.51 (Win64) PHP/7.4.26 Server at localhost Port 80*

Pour pallier ce problème, on peut désactiver le module autoindex s'il est activé en tapant la ligne suivante :

```
a2dismod autoindex
```

(Ne pas oublier de redémarrer, comme à chaque fois, le service apache2)

```
systemctl restart apache2
```

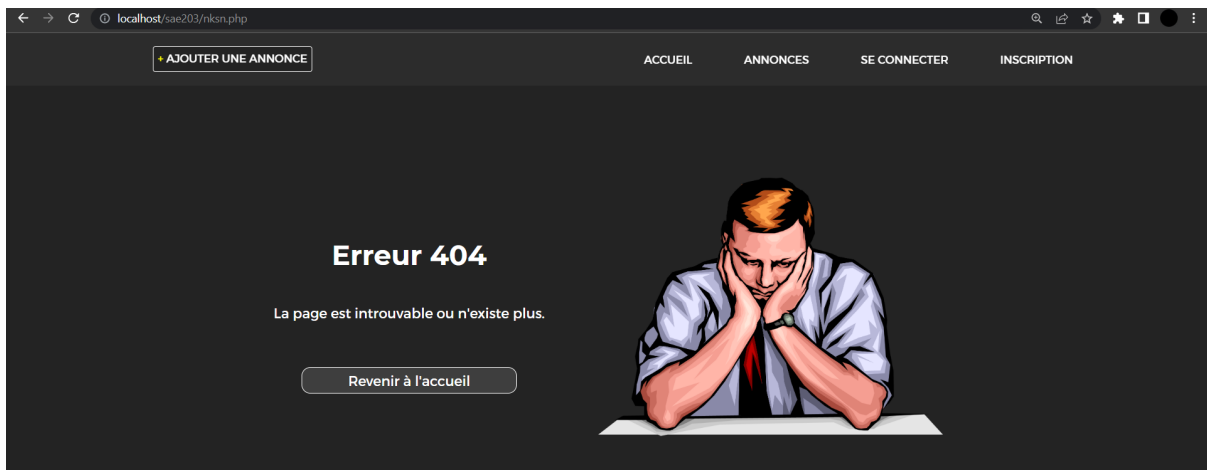
Personnellement, nous avons opté pour une autre façon afin de contourner le passage via le module ou via les configurations qui pourraient engendrer des erreurs (ex.: erreur 500) en créant un index.php dans chaque dossier qui, lorsqu'un utilisateur tente d'accéder directement à un dossier, le redirige vers la page d'accueil du site. On y gagne aussi en maintenabilité dans le cas où on déplace tous nos fichiers vers un autre serveur.

## Personnaliser les pages d'erreurs

Lorsqu'on tente d'accéder à une page web qui n'existe nul part dans la hiérarchie, on est redirigé vers une page d'erreur 404. Pour la personnaliser, on passe là-encore dans par un des fichiers configuration d'Apache *httpd.conf* pour y placer la ligne suivante :

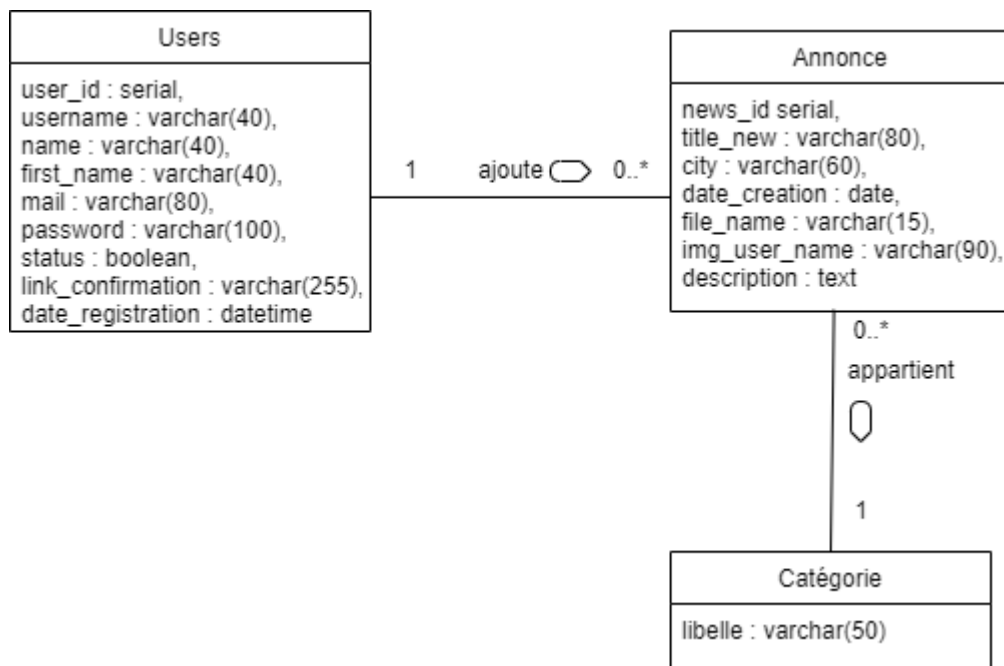
```
ErrorDocument 404 "/sae203/error404.php"
```

À présent, l'utilisateur tentant d'accéder à une page inexistante sera automatiquement redirigé par le serveur vers cette page d'erreur personnalisée.



# Base de données

## Conception



Avant d'implémenter la base de données, nous avons conçu ce diagramme de classes :

On crée trois bases, une pour les utilisateurs, une pour les annonces, et une pour les catégories. On considère qu'un utilisateur peut ajouter autant d'annonces qu'il le souhaite. Ces annonces doivent obligatoirement être liées à un id d'un utilisateur et à

une catégorie. Ces catégories seront créées directement par l'administrateur depuis la page `/sae203/private.admin.php`.

## Implémentation

Ici, on présente l'implémentation de la base de données que nous avons conçue.

Cette base de données, respectivement nommée *sae203\_bdd*, nous a été utile pour l'utilisation de notre site internet à travers notre serveur apache.

Nous avons pu l'utiliser à travers nos différents script php.

```
create database if not exists sae203_bdd;
use sae203_bdd;

drop table if exists _users;
> create table _users (
    user_id serial,
    name varchar(40) not null,
    first_name varchar(40) not null,
    username varchar(40) not null,
    mail varchar(80) not null,
    password varchar(100) not null,
    status boolean default false not null,
    link_confirmation varchar(255),
    date_registration datetime not null,
    constraint pk_users primary key (user_id),
    unique key (mail),
    unique key (username)
- );

drop table if exists _annonce;
> create table _annonce (
    news_id serial,
    title_new varchar(80) not null,
    city varchar(60) not null,
    date_creation date not null,
    file_name varchar(15) not null,
    img_user_name varchar(90) not null,
    description text not null,
    libelle varchar(50) not null,
    user_id integer not null,
    constraint pk_users primary key (news_id)
- );

drop table if exists _categorie;
> create table _categorie (
    libelle varchar(50),
    constraint pk_categorie primary key (libelle)
- );
```

# PHP

## Glossaire des fonctions

- *include* '<chemin/vers/fichier>' → Importe tout le contenu du fichier en paramètre vers la page dans laquelle est écrit cette ligne.  
Cette fonction est très utile pour importer du code php ou du code html (head, navbar, footer...) sans devoir le dupliquer partout, ce qui permet d'éviter par ailleurs le besoin de modifier chaque fichier un par un lorsqu'une modification doit s'opérer.
- *isset*(<var>) → Vérifie si une variable existe.
- *fetch*() → Retourne un tableau.  
Elle nous a été très utile dans le cas où on doit récupérer des données depuis une table.
- *fetchAll*() → Retourne elle aussi un tableau.  
Nous l'avons utilisé dans les cas où nous devons utiliser le résultat de ce tableau pour des boucles *for each* ([...]) ou *while* ([...]).
- *session\_start*() → Permet de lancer une session (ou de récupérer les données de `$_SESSION` si une session existait déjà auparavant) et d'initialiser la superglobale `$_SESSION` disponible.
- *session\_destroy*() → Détruit toutes les données enregistrées dans la superglobale `$_SESSION` et met fin à la session active s'il y en a une.
- *header*('location: <chemin/vers/fichier>') → Permet de faire une redirection.  
Il faut le succéder d'un *exit* ou d'un *die*() afin de stopper le script.
- *empty*([...]) → Vérifie si une variable ou un tableau est vide.
- *substr*(<string>,<index>) retourne les caractères d'un *string* après l'index spécifié.

- `strtolower(<string>)` → retourne la chaîne de caractères paramétrées en minuscules.
- `strcmp(<string>, <string>)` → Compare deux chaînes de caractères
- `echo()` → Affiche la chaîne de caractère inscrite dans l'encadré
- `n12br([...])` → Permet de récupérer les retours à la ligne.
- `htmlentities([...])` → Permet de convertir des caractères en entités HTML et de sécuriser les champs de saisie.
- `sha1` -> Permet d'encrypter une chaîne en sha1.  
On l'utilise pour les encrypter les mots de passe. En cas d'attaque de la base de données, l'attaquant ne pourra pas voir les mots de passe en brut.
- `trim([...])` → Supprime les espaces (ou d'autres caractères) en début et fin de chaîne
- `filter_var(<string>, FILTER_VALIDATE_EMAIL)` → Renvoie un booléen qui vérifie, pour ce cas avec ce paramètre, si le <string> à la syntaxe d'une adresse email.
- `bin2hex(random_bytes(<int>))` → Génère x caractères aléatoires.  
Cette fonction nous a été utile pour générer le code correspondant au lien de confirmation.
- `explode(<delimiter>, <string>)` → Retourne un tableau de chaînes séparées par le délimiteur.
- `$result = $connexion -> prepare(<requête_sql>)` → Permet d'initialiser une requête préparée (requête avec des '?' à la place de un ou plusieurs arguments)
- `$result = execute(array([...]))` → Exécute la requête préparée en utilisant un `array()` qui remplacera les '?' de cette requête par les éléments de ce tableau.



## Connexion à la base (PDO)

```
class connexionDB {
    private $host      = 'localhost';    // nom de l'host
    private $name       = 'sae203_bdd';   // nom de la base de donnée
    private $user       = 'root';         // utilisateur
    private $password   = 'lannion';      // mot de passe
    private $connexion;

    function __construct($host = null, $name = null, $user = null, $password = null){
        if($host != null){
            $this->host = $host;
            $this->name = $name;
            $this->user = $user;
            $this->password = $password;
        }
        try{
            $this->connexion = new PDO('mysql:host=' . $this->host . ';dbname=' . $this->name,
                $this->user, $this->password, array(PDO::MYSQL_ATTR_INIT_COMMAND =>'SET NAMES UTF8',
                PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING));
        }catch (PDOException $e){
            echo 'Erreur : Impossible de se connecter à la BDD !';
            die();
        }
    }

    public function query($sql, $data = array()){
        $req = $this->connexion->prepare($sql);
        $req->execute($data);
        return $req;
    }

    public function insert($sql, $data = array()){
        $req = $this->connexion->prepare($sql);
        $req->execute($data);
    }
}

// Faire une connexion à votre fonction
$link = new connexionDB();
```

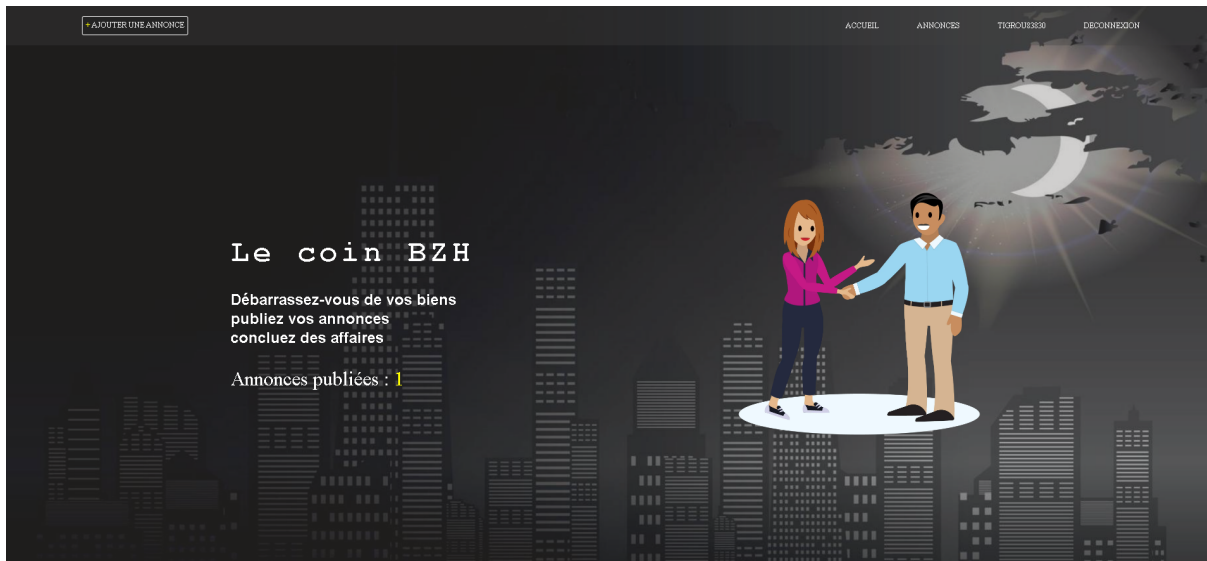
Concernant la partie connexion, on utilise pour cela une classe php. On renseigne dans un premier temps les données de connexion, puis on établit avec ces données la connexion a PDO. Si elle ne fonctionne pas, on renvoie une erreur.

Nous avons choisi PDO au lieu de MySQLi car les fonctions implémentées sont bien plus simples à utiliser, notamment avec les requêtes préparées par exemple.

Ensuite, nous avons créé deux fonctions.

- query() → qui retourne le résultat de la requête dans le cas d'un select.
- insert() → qui exécute une requête mais qui ne retourne rien dans le cas d'un insert, update, ou delete.

## Page d'accueil



Nous avons implémenté dans cette page d'accueil une requête qui récupère le nombre d'annonces postées sur le site afin de l'afficher grâce à un *select*. Si la requête ne renvoie rien, alors il n'y a pas d'annonce donc on lui attribue la valeur 0. Sinon, on lui attribue la valeur du *count(\*) nb* présent dans la requête.

PHP :

```
session_start();
include 'private/connexionbdd.php';
$req_get_nb_annonces = $LINK -> query("select count(*) nb from _annonce",
    array());
$req_get_nb_annonces = $req_get_nb_annonces -> fetch();
if (!$req_get_nb_annonces['nb'])
{
    $nb_annonces = 0;
} else{
    $nb_annonces = $req_get_nb_annonces['nb'];
}
```

HTML:

```
<p class="acc_count">Annonces publiées : <span class="yellow"><?php echo $nb_annonces ?></span></p>
```

## Inscription

Connectez-vous'." data-bbox="125 115 885 363"/>

Pour la page inscription, on demande à l'utilisateur de remplir tous ces champs : nom, prénom, nom d'utilisateur, email, mot de passe et une confirmation de mot de passe.

À l'envoi du formulaire, on vérifie plusieurs conditions, et on affiche une erreur en cas de besoin :

- Le nom d'utilisateur ne doit pas déjà existé
- L'email ne doit pas déjà existé
- Le mot de passe et la confirmation du mot de passe doivent être égales
- La longueur du mot de passe doit avoir un minimum de six caractères
- Tous les champs doivent être renseignés.

on sécurise les données `$_POST`, exceptées les mots de passe qui sont cryptées en *SHA1*,

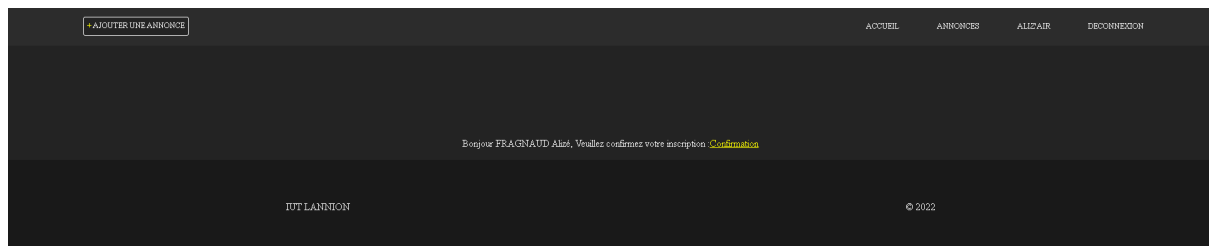
avec un *htmlentities* pour empêcher toute tentative d'insérer un script et un *trim* pour effacer les potentiels espaces au début et à la fin de chaque chaîne

Si toutes les conditions réunies sont propices à l'insertion de ce nouvel utilisateur, on passe à la dernière étape d'enregistrement :

On insère dans un premier temps les premières données (nom, prénom, nom d'utilisateur, mot de passe, statut=*false*, date d'inscription=*now()*), puis on récupère son id créé automatiquement et on initialise une variable qui correspond au code du lien de confirmation ayant comme valeur l'id + 25 caractères aléatoires. Enfin on exécute un *update* sur le lien de confirmation pour le définir.

L'insertion étant terminé, on connecte l'utilisateur temporairement à l'aide de la superglobale `$_SESSION (id, nom, prenom, username, lien)` et on le redirige vers `lien_confirmation.php` qui le déconnectera après avoir récupéré des données utiles pour la confirmation du lien.

## Lien de confirmation



Après l'inscription, on continue sur la partie du lien de confirmation.

Dans `lien_confirmation.php`, on récupère dans une variable le code servant au lien de confirmation et on détruit la session.

Dans le lien renseigné dans le html, on utilise la variable du lien pour l'insérer dans l'id du lien de confirmation.

Lorsque l'utilisateur clique sur ce lien de confirmation, le script `confirmation.php` est lancé. Celui récupère l'id de l'url puis vient vérifier si un utilisateur dont le code du lien est égal à cet id existe. Si c'est le cas, alors on lui exécute un `update` qui supprime son code de confirmation et qui actualise `status` à `true`.

Source : `lien_confirmation.php`

PHP:

```
else{
    $lien_confirmation_inscription=$_SESSION['lien'];
}
session_destroy();
```

HTML:

```
<div class="flex bloc_link_conf">
  <p>Bonjour <?php echo $_SESSION['nom'] . " " . $_SESSION['prenom']?>, Veuillez confirmer votre inscription : </p>
  <a class="link yellow" href="confirmation.php?id=<?php echo $lien_confirmation_inscription; ?>">Confirmation</a>
</div>
```

Source : `Confirmation.php`

php :

```

else{
    $id_url = $_GET['id'];
    $sql_verif = $LINK->query("select count(*) nb from _users where link_confirmation = ?",
    array($id_url));
    $sql_verif = $sql_verif -> fetch();
    if($sql_verif['nb'] == 0)
    {
        header('Location: index.php');
        exit;
    }
    else{
        $sql_confirm = $LINK->insert("update _users set link_confirmation = null, status = true where link_confirmation = ?;",
        array($id_url));
        header("location: login.php");
        exit;
    }
}
}

```

## Connexion de l'utilisateur

Pour la page connexion , on demande à l'utilisateur de renseigner son adresse email ainsi que son mot de passe. À l'envoi de ce formulaire, on vérifie là-encore la véracité d'autres conditions :

- L'email doit exister dans la base.
- Le mot de passe doit correspondre à celui de l'email.
- Le compte doit être vérifié (status = true).
- La longueur du mot de passe doit être supérieure à cinq.

Une fois ces conditions remplies, le script récupère les données reliées à cette utilisateur et enregistré une partie de celles-ci dans la superglobale \$\_SESSION.

On réitère la même manipulation concernant la sécurisation du champ email et l'encrytage du mot de passe.

Source : login.php

PHP:

```
// Si toutes les conditions sont remplies alors on fait le traitement
if ($valid) {
    $user=$LINK->query("select user_id, name, first_name, mail, username from _users where mail = ?",
        array($mail));
    $user = $user -> fetch();

    $_SESSION['id'] = $user['user_id'];
    $_SESSION['nom'] = $user['name'];
    $_SESSION['prenom'] = $user['first_name'];
    $_SESSION['mail'] = $user['mail'];
    $_SESSION['username'] = $user['username'];

    header('Location: index.php');
    exit;
}
```

HTML:

```
<div class="form_connexion_reg">
<h1>CONNEXION</h1>
<form class="form_connexion" method="post" action="">

    <div class="flex">
        <p class="label_input">Email<span class="required">*</span> :</p>
        <input type="email" maxlength="78" placeholder="Adresse mail" name="mail" autocomplete="off" value="<?php if(isset($mail)) echo $mail; ?>" required>
    </div>

    <div class="flex">
        <p class="label_input">Mot de passe<span class="required">*</span> :</p>
        <input type="password" minlength="6" placeholder="Mot de passe" name="mdp" autocomplete="off" required>
    </div>

    <br>
    <button type="submit" minlength="6" class="submit_connexion" name="connexion">Envoyer</button>
    <br>
    <div class="error"><?php echo $message_error; ?></div>
</form>
</div>
```

## Déconnexion de l'utilisateur

Lorsqu'un utilisateur désire se déconnecter, il lui suffit de cliquer sur le lien de déconnexion accessible depuis n'importe quelle page si on est connecté. On détruit sa session à l'aide du `session_destroy()` puis on le redirige vers la page d'accueil.

Source : deconnexion.php

PHP :

```
<?php
session_start();
session_destroy();
header('Location: /sae203/index.php'); // Ici il faut mettre la page sur lequel l'utilisateur sera redirigé.
exit;
?>
```

Source : tous (si connecté)

## Sessions et redirections automatiques

On a vu précédemment qu'on exécutait du PHP pour l'inscription, la connexion, etc. Or on doit pouvoir gérer côté backend les redirections afin qu'un utilisateur ne puisse pas accéder à la page d'inscription ni à la page de connexion, et qu'un utilisateur anonyme ne puisse pas ajouter d'annonce.

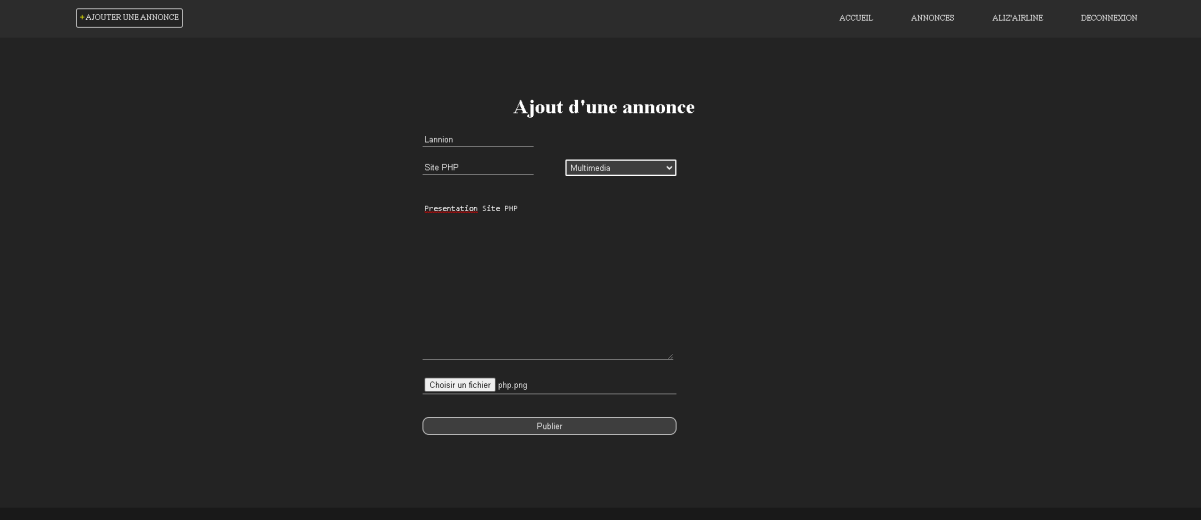
Exemple ci-contre:

Si une session est active, l'utilisateur est redirigé vers index.php.

```
if (isset($_SESSION['id'])) {  
    header('Location: index.php');  
    exit;  
}
```

## Gérer les annonces

### Créer une annonce



The screenshot shows a web application interface for adding an announcement. At the top, there is a navigation bar with a button '+ AJOUTER UNE ANNONCE' on the left and links 'ACCUEIL', 'ANNONCES', 'ALIZABLINE', and 'DECONNEXION' on the right. The main content area is titled 'Ajout d'une annonce'. It contains a form with the following fields: 'Nom' (text input), 'Site PHP' (text input), 'Multimedia' (dropdown menu), and 'Presentation Site PHP' (text input). Below these fields is a file upload section with a button 'Choisir un fichier' and a file name 'file.png'. At the bottom of the form is a 'Publier' button.

Pour ajouter une annonce, on vérifie si une session existe et on demande de remplir un titre, une ville, une description, et de choisir une catégorie ainsi qu'un fichier. À l'envoi des données pour une insertion, on vérifie quelques conditions :

- Tous les champs doivent être remplis
- Le fichier doit respecter certains formats (jpg, jpeg, png, webp)
- Le fichier ne doit pas dépasser 2Mo. (Afin de ne pas saturer l'espace avec des images en 4K par exemple)
- Que la catégorie choisie n'est pas celle par défaut.

Lorsque la plupart de ces conditions sont vérifiées, on commence par traiter le cas de l'image. Tout d'abord, on récupère son extension afin de le comparer avec les valeurs dans le array() des formats acceptés. Puis, s'il l'extension est présente, on la récupère dans une variable. Ensuite, on définit un nom d'image unique : ici, on en crée un en donnant l'id de l'utilisateur suivi d'un caractère suivi d'un entier équivalent au nombre d'images déjà créées. Ce chemin unique est nécessaire pour éviter tout type d'écrasement de fichier dans le cas où on garderait le nom de l'image donnée par l'utilisateur.

Puis, on place l'image dans le bon répertoire avec le nom unique généré auparavant. Si le déplacement est un succès, on opère l'insertion de l'annonce dans la base à l'aide d'une requête préparée.

Source : déposer-annonce.php

PHP:

```
else {
    $req_cat = $LINK->query("SELECT libelle FROM _categorie",
        array());
    $req_cat = $req_cat->fetchAll(PDO::FETCH_ASSOC);

    $req_nb_news = $LINK->query("SELECT count(*) nb from _annonce",
        array());
    $req_nb_news = $req_nb_news->fetch();

    $message_error = ' ';

    if (isset($_FILES['image']) && !empty($_FILES['image']['name'])) {
        $title_new = htmlentities($_POST['title_new']);
        $city = htmlentities($_POST['city']);
        $description = htmlentities($_POST['description']);
        $categorie = htmlentities($_POST['categorie']);
        $tailleMax = 2097152;
        $extensionValides = array('jpg', 'jpeg', 'png', 'webp');

        if (strcmp($categorie, 'default') === 0) {
            $message_error = 'Veuillez choisir une catégorie';
        } elseif ($_FILES['image']['size'] <= $tailleMax) {
            $extensionUpload = strtolower(substr(strrchr($_FILES['image']['name'], '.'), 1));
            if (in_array($extensionUpload, $extensionValides)) {
                $nom_unique_img = $_SESSION['id'] . "a" . $req_nb_news['nb'] . $extensionUpload;
                $chemin = "../ressources/img/dynamic/" . $nom_unique_img . "." . $extensionUpload;
                $deplacement = move_uploaded_file($_FILES['image']['tmp_name'], $chemin);
                if ($deplacement) {
                    $insertImage = $LINK->insert("insert into _annonce (title_new, city, date_creation, file_name, img_user_name, description, libelle, user_id) values (?, ?, ?, ?, ?, ?, ?, ?)",
                        array($title_new, $city, $nom_unique_img, $_FILES['image']['name'], $description, $categorie, $_SESSION['id']));
                } else {
                    $msg = "Impossible d'importer l'image";
                }
            } else {
                $msg = "Mauvais format (jpg, png, jpeg, webp)";
            }
        } else {
            $msg = "Votre image ne doit pas dépasser 2Mo";
        }
    }
}
```



## HTML :

```
<h1 class="text-center">Ajout d'une annonce</h1>
<form action="" method="post" enctype="multipart/form-data">
  <input type="text" maxlength="58" placeholder="Ville" name="city" autocomplete="off" value="<?php if(isset($city)) echo $city; ?>" required>
  <br>
  <div class="multi-flex flex">
    <input type="text" maxlength="78" placeholder="Titre" name="title_new" autocomplete="off" value="<?php if(isset($title_new)) echo $title_new; ?>" required>
    <select name="categorie" id="categorie">
      <option value="default">- Catégorie -</option>
      <?php
        foreach($req_cat as $row) {
          echo '<option value="' . $row['libelle'] . '">' . $row['libelle'] . '</option>';
        }
      <?php
    </select>
  </div>
  <br>
  <textarea placeholder="Description..." maxlength="2000" name="description" autocomplete="off" value="<?php if(isset($description)) echo $description; ?>" required>
  <br>
  <input type="file" id="avatar" name="image" accept="image/png, image/jpeg, image/webp, image/jpg" required>
  <br>
  <?php echo '<p class="error">' . $message_error . '</p>?>'
  <button class="submit_connexion" type="submit" name="add_new">Publier</button>
</form>
```

## Recherche les annonces avec une catégorie

Afin de rechercher une annonce, l'utilisateur doit sélectionner une catégorie puis cliquer sur rechercher. On doit traiter un cas spécial pour ce traitement. Si l'utilisateur ne sélectionne pas de catégorie, alors toutes les annonces s'affichent, sinon, seules les annonces correspondant à la catégorie choisie par l'utilisateur doivent s'afficher. Pour cela, on récupère chacune des annonces dans un *select* où la catégorie est égale à celle sélectionnée par l'utilisateur, puis on le redirige vers `admin.php` puisque le *action* dans le formulaire faisait lien avec un fichier extérieur au dossier courant.

Source : `rechercher_categorie.php`

## PHP

```
if (isset($_POST['research_cat']) && (strcmp($_POST['categorie'], 'default') != 0)){
    $categorie = $_POST['categorie'];
    $req_get_new_filter = $LINK->query("select * from _annonce natural join _users where libelle = ? order by date_creation desc",
        array($categorie));
    $req_get_new_filter = $req_get_new_filter->fetchAll(\PDO::FETCH_ASSOC);

    header('location: /sae203/private/admin.php');
    exit;
} else {
    header('location: /sae203/private/admin.php');
    exit;
}
```

Source : annonces.php

HTML :

```
<form class="filtrage_annonce flex" action="" method="post">
  <select name="categorie" id="categorie">
    <option value="default">- Catégorie -</option>
    <?php
      foreach($req_get_cat as $row) {
        echo '<option value="' . $row['libelle'] . '>' . $row['libelle'] . '</option>';
      }
    ?>
  </select>
  <button type="submit" class="submit_connexion" name="research_cat">Rechercher</button>
</form>
```

Supprimer une annonce



Pour supprimer une annonce, l'admin doit seulement cliquer sur la croix en haut à droite de la page. Pour cela, on récupère dans la *value* du bouton le numéro de l'annonce ainsi que le numéro de l'utilisateur auquel elle appartient.

Ensuite, on sépare en deux variables les deux données grâce au *explode()* puis on exécute un *delete* sur la table des annonces où l'id de l'annonce et l'id de l'utilisateur correspondent et permettent d'identifier l'annonce concernée.

Enfin, une redirection vers admin.php est réalisée.

Source :

PHP : suppr\_annonce.php

```

if (isset($_POST['suppr_annonce'])){
    $suppr_annonce = $_POST['suppr_annonce'];
    $pieces = explode(";", $suppr_annonce);
    $news_id = $pieces[0];
    $user_id = $pieces[1];

    $req_suppr_annonce = $LINK->insert("delete from _annonce where news_id = ? and user_id = ?",
        array($news_id, $user_id));

    header('location: /sae203/private/admin.php');
    exit;
} else{
    header('location: /sae203/index.php');
    exit;
}
?>

```

Source : admin.php

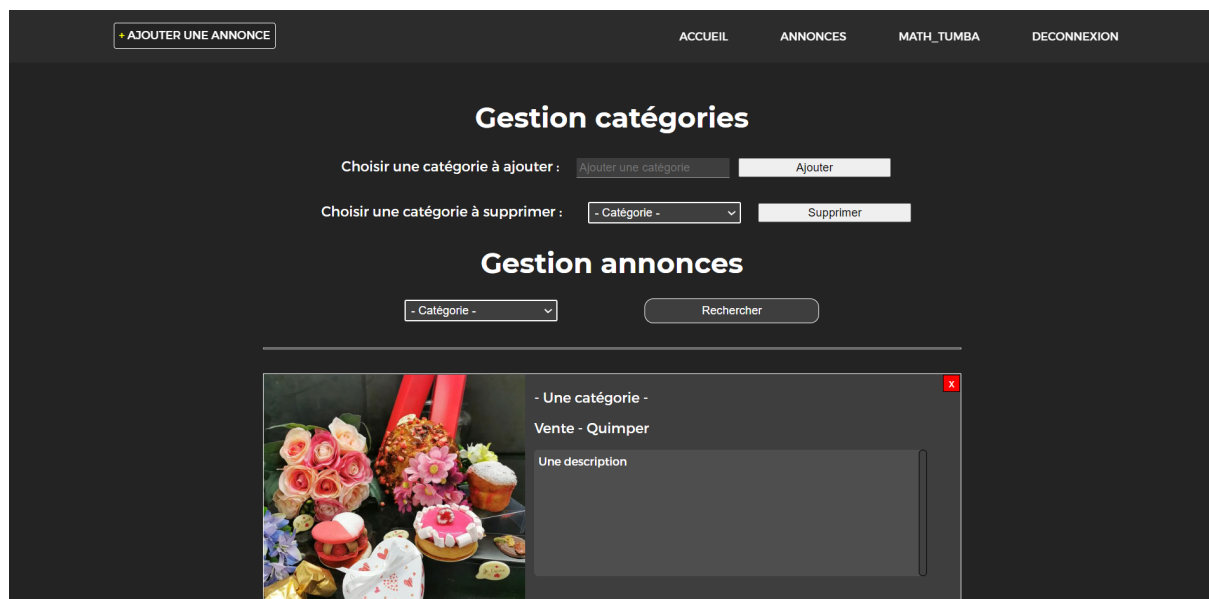
HTML :

```

<form action="../../ressources/libs/suppr_annonce.php" method="post">
    <button type="submit" class="suppr" name="suppr_annonce" value="'. $row['news_id'] . ';' . $row['user_id'] .'">x</button>
</form>
<div class="infos_annonce">

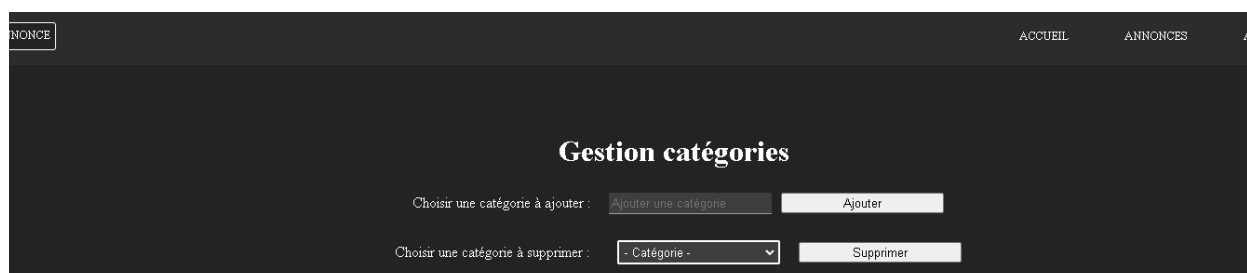
```

## Page administrateur



Cette page administrateur est bloquée aux utilisateurs par une authentification Apache2 si l'on a configuré le directory dans les configurations (CF : Système de connexion à une page d'administrateur). Une fois authentifié, on peut créer / supprimer des catégories et supprimer n'importe quelle annonce.

## Gérer les catégories



### Créer une catégorie

Pour créer une catégorie, on récupère les données du champ envoyé et on vérifie qu'il n'existe pas déjà dans la base des catégories. Si cette condition est remplie, alors la catégorie est ajoutée dans la base de données.

Source : ajout\_categorie.php

PHP:

```
<?php
include '../private/connexionbdd.php';
if(isset($_POST['add_cat'])){
    $categorie = $_POST['categorie_input'];
    $req_sel_cat = $LINK->query("select libelle from _categorie where libelle = ?",
        array($categorie));
    $req_sel_cat = $req_sel_cat -> fetch();

    if(!isset($req_sel_cat['libelle'])){
        $req_add_cat = $LINK->insert("insert into _categorie (libelle) values(?)",
            array($categorie));
    } else {
        header('location: /sae203/private/admin.php?error=21');
        exit;
    }
    header('location: /sae203/private/admin.php');
    exit;
}
```

HTML :

```
<form class="flex" action="../ressources/libs/ajout_categorie.php" method="post">
    <label>Choisir une catégorie à ajouter : </label>
    <input type="text" placeholder="Ajouter une catégorie" name="categorie_input" autocomplete="off" value="" required>
    <button type="submit" class="submit_admin" name="add_cat">Ajouter</button>
</form>
```

### Supprimer une catégorie

Pour supprimer une catégorie, l'utilisateur doit sélectionner une catégorie présente dans la liste déroulante. Plusieurs conditions / cas sont traités.

- L'utilisateur ne peut pas supprimer une catégorie si au moins une annonce fait partie de cette catégorie.

- Rien n'est supprimé lorsque l'utilisateur clique sur supprimer sans avoir sélectionner de catégorie.

Si ces cas sont traités, alors la catégorie sélectionnée est supprimée de la base de données.

Source : supprimer\_categorie.php

PHP:

```

<?php
include '../private/connexionbdd.php';

if(isset($_POST['delete_cat'])){
    $categorie = $_POST['categorie'];
    if (strcmp($categorie, "default") === 0){
        header('location: /sae203/private/admin.php?error=11');
        exit;
    } else {
        $req_cat_verif_del = $LINK->query("select count(*) nb from _annonce where libelle = ?",
            array($categorie));
        $req_cat_verif_del = $req_cat_verif_del -> fetch();

        if ($req_cat_verif_del['nb'] > 0){
            header('location: /sae203/private/admin.php?error=12');
            exit;
        } else {
            $req_del_cat = $LINK->insert("delete from _categorie where libelle = ?",
                array($categorie));
        }
    }
    header('location: /sae203/private/admin.php');
    exit;
} else{
    header('location: /sae203/index.php');
    exit;
}
?>
```

HTML :

```
<form class="flex" action="../../../ressources/libs/supprimer_categorie.php" method="post">
  <label>Choisir une catégorie à supprimer : </label>
  <select name="categorie" id="Categorie">
    <option value="default">- Catégorie -</option>
    <?php
      if(isset($_POST['delete_cat'])){
        foreach($req_get_cat as $row) {
          if ((strcmp($row['libelle'], $categorie) !== 0)){
            echo '<option value="' . $row['libelle'] . '">' . $row['libelle'] . '</option>';
          }
        }
      }
      elseif(isset($_POST['add_cat'])){
        if (!in_array($categorie, $row)){
          echo '<option value="' . $categorie . '">' . $categorie . '</option>';
        }
        foreach($req_get_cat as $row) {
          echo '<option value="' . $row['libelle'] . '">' . $row['libelle'] . '</option>';
        }
      } else if (!isset($_POST['add_cat']) && !isset($_POST['del_cat'])){
        foreach($req_get_cat as $row) {
          echo '<option value="' . $row['libelle'] . '">' . $row['libelle'] . '</option>';
        }
      }
    <?>
  </select>
  <button type="submit" class="submit_admin" name="delete_cat">Supprimer</button>
  <br>
</form>
```

# CRON

Le CRON sert à créer une sorte de planificateur des tâches dont les paramètres sont une date/heure précise. On peut donc le programmer afin de lui donner un comportement. Dans notre cas, on le programme afin qu'il exécute un script tous les jours à 3h du matin, supprimant tous les utilisateurs qui ont un *status = false* (qui n'ont pas cliqué sur le lien de confirmation lors de l'inscription).

On commence par trouver le chemin vers php :

```
whereis php → /usr/bin/php7.4
```

On ajoute le cron :

```
cron -e
```

Puis on rentre la ligne suivante :

```
0 3 * * * /usr/bin/php7.4  
/var/html/sae203/private/cron_delete.php
```

```
<?php  
include '/sae203/private/connexionbdd.php';  
$req_delete_all_temp = $LINK->insert("delete * from _users where status = false",  
    array());  
?>
```

Code php exécuté (/sae203/private/cron\_delete.php) à 3h du matin

# Approfondissements possibles

Si nous avions eu plus de temps, nous aurions pu développer :

- les medias queries afin de rendre le site responsive
- une page personnelle pour chaque utilisateur avec le pouvoir de modifier ses informations et de supprimer ses et seulement ses propres annonces
- le chargement par page d'annonces plutôt que d'afficher tout sur une même page, ce qui est contraignant lorsque le nombre d'annonces postées dépasse la trentaine.
- un système de recherche plus poussée pour trouver les annonces (repérage de mot-clé dans les titres et descriptions avec un système d'importance et de priorité)



# BILAN

Pour conclure, nous pouvons dire que c'était un projet intéressant mais malheureusement, nous avons jugé que nous avions trop peu de temps pour réaliser notre site, nous aurions pu intégrer d'autres fonctionnalités encore plus intéressantes.. De plus, nous aurions aimé avoir quelques cours encadrés de PHP orienté web à l'IUT qui nous auraient permis de réaliser notre site avec plus de facilités dès le commencement du développement.

# Bibliographie

<https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql>

<https://www.mysqltutorial.org/mysql-datetime/>

<https://stackoverflow.com/>

<https://www.php.net/manual/fr/>

<https://www.codexworld.com/>

<https://www.w3schools.com/php/>

// lien Site

//lien Glossaire