



# RAPPORT DE STAGE

du 30 mars au 26 juin 2020

Stagiaire : Jade Guehoun | Tutrice : Mme GONDALLIER Audrey  
Formateur.rices : Mme JOLY Marion et Mr BERNARD Hervé



## Sommaire

<b>Remerciements</b>	PAGE 4
<b>Introduction</b>	PAGE 6
<b>Projets réalisés</b>	PAGE 7-42
I/ lesbavardes.org	PAGE 7-35
A.    Présentation du collectif	PAGE 7
B.    Partir de l'existant	PAGE 7-8
C.    Gestion du projet	PAGE 8
D.    Back-End	PAGE 8-35
1.    Technologies utilisées	PAGE 8-10
2.    Premiers pas avec CodeIgniter	PAGE 10-11
3.    Version multilingue	PAGE 12-15
4.    Création de session	PAGE 16-24
5.    Réalisation d'un CRUD	PAGE 25-35
E.    Mise en ligne	PAGE 35
II/ femmes-citoyennes.fr	PAGE 36-42
A.    Présentation du collectif	PAGE 36
B.    Technologies utilisées	PAGE 36-37
C.    Création du site avec WordPress	PAGE 38-42
D.    Mise en ligne	PAGE 42
<b>Conclusion</b>	PAGE 44



## Remerciements

Je tiens à remercier l'ensemble des intervenants ayant contribué à ce que cette période de stage soit, malgré la période de confinement et cette crise sanitaire sans précédents, un moment de partage, de bienveillance et de patience quant à mes besoins et interrogations.

Ces remerciements s'adressent tout d'abord aux membres de l'équipe pédagogique du centre de l'AFPA d'Amiens. Leurs conseils m'ont été d'une aide précieuse tout au long de mon parcours en centre. Leurs enseignements, théoriques et pratiques, m'ont été indispensables pour mener à terme cette formation.

Je tiens également à remercier Madame Audrey Gondallier, coprésidente de l'association *les Bavardes*, qui, en acceptant de me compter parmi ses effectifs, m'a permis d'effectuer mon stage aux côtés de son équipe développement.

Merci à Lucille Lepers, Noémie Claude et Marion Joly qui ont su, pendant ces 12 dernières semaines me conseiller et me guider afin que je puisse mener à bien les projets qui m'ont été confiés.



## Introduction

Le lundi 30 mars 2020 j'ai intégré l'équipe développement de l'association *les Bavardes* pour la réalisation de mon stage en vue de l'obtention du diplôme de développeur web web mobile dans le cadre de la formation mise en place par l'AFPA d'Amiens.

Ce choix de formation a été pour moi un moyen de pouvoir allier curiosité et découverte. Le domaine de l'informatique m'a toujours intrigué, désormais je peux dire que cet univers me passionne.

Dans le cadre de ma formation pour le titre professionnel de développeur web web mobile, je vais vous présenter deux projets que j'ai réalisés durant mon stage aux côtés du collectif *les Bavardes*.

Mon premier projet est la refonte d'un site existant, [lesbavardes.org](http://lesbavardes.org). Ce site était exclusivement fait d'HTML, CSS et Bootstrap. J'ai été chargé de la création de la partie back-office. J'ai donc créé toutes les bases de données du site, des CRUD, une session admin, un onglet et une page pour les événements. J'ai également été chargé de créer une version anglaise du site. J'ai choisi de faire ce projet à l'aide du MVC CodeIgniter.

Mon second projet a été de créer un site vitrine avec WordPress, pour « *La fausse campagne aux vraies revendications* » du collectif *Citoyennes Amiénoises Féministes*.

# Projets réalisés – lesbavardes.org

## A - Présentation du collectif

Les Bavardes est un collectif lesbien et féministe qui existe sur Amiens depuis juin 2017. C'est aussi un collectif bi, trans, queer et hétéro, engagé pour la visibilité de toutes les femmes au travers d'espaces et d'actions culturelles et artistiques, ouverts à toutes et à tous, visant à agir, se rencontrer, s'interroger et faire la fête. Ce collectif est à la tête de nombreuses mobilisations, actions de sensibilisation pour lutter contre tous types d'inégalités, pour lutter contre le sexisme, la lesbophobie, la transphobie et le racisme.

## B – Partir de l'existant



L'une des raisons principales de mettre en place ce site web est de rendre plus visible le collectif auprès des financeurs, présenter les différentes actions et collaborations, participer aux archives féministes.

Sur ce site n'existait que la partie front-end.

Qu'est-ce que le front-end ?

Le front-end est la partie du code qui est reçue par le client. Le client est notre navigateur Web. Il s'agit des éléments du site web que l'on aperçoit à l'écran et avec lesquels on pourra interagir.

Notre navigateur va afficher nos pages web. Le code client que notre navigateur peut comprendre est composé de 3 langages différents : HTML, CSS et JS.

Sur la version 1.0 le site était exclusivement fait des 3 langages suivants :

### HTML (Hyper Text Markup Language)

HTML est un langage composé de tags, de balises en français. Il va nous permettre de représenter la structure, le squelette de nos pages web.



## CSS (Cascading Style Sheets)

CSS est un langage qui va mettre en forme nos pages web et les décorer. Il va désigner nos éléments HTML à l'aide de sélecteurs et va appliquer le style CSS.

```
h2 {  
  font-family: 'Quicksand', cursive;   
  color: black;  
}  
                                style.css  
  
<div class="col-lg-12 shadow p-3 mt-5">  
  <h2> Courrier Picard </h2>  
  <ul>  
    <li>
```

## Bootstrap

Bootstrap est un framework HTML, CSS et JS. C'est une structure qui contient de nombreux composants prêts à l'emploi : boutons, listes déroulantes... Bootstrap est surtout connu et indispensable pour développer « responsive ».

Que signifie « responsive » ?

C'est tout d'abord une façon d'appréhender la conception d'un site Web. Notre site devient une page flexible qui s'adapte automatiquement à la taille de l'écran (smartphone, tablette, ordinateur).



## C – Gestion du projet

La gestion du projet s'est faite à l'aide de Github. Github est un service en ligne qui permet d'héberger ses dépôts de code. C'est un logiciel de gestion de version, ce qui signifie qu'il gère les modifications d'un projet sans en écraser toutes les parties.

## D – Back-end

Qu'est-ce que le back-end ?

C'est l'interface invisible à l'utilisateur qui regroupe toutes les manipulations opérées par un administrateur (c'est-à-dire la gestion de notre site internet).

### 1. Technologies utilisées

Pour la création de la partie back-office du site lesbavardes.org, j'ai utilisé les technologies suivantes :

## **Visual Studio Code**

Visual Studio Code est un éditeur de code open-source développé par Microsoft. Visual Studio Code est développé avec Electron et exploite des fonctionnalités d'édition avancées du projet Monaco Editor. Principalement conçu pour le développement d'application avec JavaScript, TypeScript et Node.js, l'éditeur peut s'adapter à d'autres types de langages grâce à un système d'extension bien fourni.

## **Wampserver**

Wampserver est une plateforme de développement web permettant de faire fonctionner localement des scripts PHP. C'est un environnement comprenant trois serveurs, un interpréteur de script, ainsi que PHPMyAdmin pour l'administration Web des bases de données.

## **PHPMyAdmin**

PHPMyAdmin est l'une des plus célèbres interfaces pour gérer les bases de données MySQL sur un serveur PHP. Cette interface permet d'exécuter des requêtes comme la création de bases de données, de tables, des requêtes d'insertions, de mises à jours, de suppressions et de modifications, ainsi que l'attribution et la révocation de droits et l'import/export.

## **CodeIgniter**

CodeIgniter est un framework php basé sur l'architecture MVC (nom donné à une manière d'organiser son code). Cette architecture permet aux pages web de contenir un minimum de script.

Le modèle représente la structure des données. En règle générale, les classes de modèles contiennent des fonctions pour récupérer, insérer, supprimer et mettre à jour les informations dans la base de données.

La vue est l'information présentée à un utilisateur. Une vue sera normalement une page web, mais dans CodeIgniter, une vue peut également être un fragment de page, tel qu'un en-tête ou un pied de page.

Le contrôleur sert d'intermédiaire entre le modèle, la vue et toute autre ressource nécessaire au traitement de la requête http et à la génération d'une page Web.

CodeIgniter a une approche assez souple du MVC puisque les modèles ne sont pas requis. On peut les ignorer et créer une application de manière minimale à l'aide de contrôleur et de vue. CodeIgniter est également une boîte à outils pour la création d'application web avec un ensemble riche de bibliothèque.

cache	15/06/2020 10:53	Dossier de fichiers	
config	15/06/2020 10:53	Dossier de fichiers	
controllers	15/06/2020 10:53	Dossier de fichiers	
core	15/06/2020 10:53	Dossier de fichiers	
helpers	15/06/2020 10:53	Dossier de fichiers	
hooks	15/06/2020 10:53	Dossier de fichiers	
language	15/06/2020 10:53	Dossier de fichiers	
libraries	15/06/2020 10:53	Dossier de fichiers	
logs	15/06/2020 11:32	Dossier de fichiers	
models	15/06/2020 10:53	Dossier de fichiers	
third_party	15/06/2020 10:53	Dossier de fichiers	
views	15/06/2020 10:53	Dossier de fichiers	
.htaccess	04/05/2020 18:42	Fichier HTACCESS	1 Ko
index	04/05/2020 18:42	Chrome HTML Do...	1 Ko

## 2. Premiers pas avec CodeIgniter

J'ai configuré la base\_url dans le fichier config.php avec le nom du domaine.

```
application > config > config.php
24 |
25 */
26 $config['base_url'] = 'http://localhost/siteBavardesdesk/';
27
```

J'ai configuré le fichier autoload.php avec les bibliothèques et assistants nécessaires.

```
$autoload['libraries'] = array('upload', 'database', 'session', 'form_validation', 'pagination');

$autoload['helper'] = array('url', 'file', 'form', 'captcha');
```

J'ai modifié le fichier route.php avec le nom du contrôleur de la page d'accueil du site lesbavardes.org.

```
wamp > www > siteBavardesdesk > application > config > routes.php
*/
$route['default_controller'] = 'accueil';
```

J'ai créé une base de données.

<table><tr><th colspan="2">bavardes_site podcast</th></tr><tr><td>podcast_id</td><td>int(11)</td></tr><tr><td>podcast_link</td><td>varchar(300)</td></tr><tr><td>podcast_name</td><td>varchar(100)</td></tr><tr><td>file_name</td><td>varchar(25)</td></tr></table>	bavardes_site podcast		podcast_id	int(11)	podcast_link	varchar(300)	podcast_name	varchar(100)	file_name	varchar(25)	<table><tr><th colspan="2">bavardes_site people</th></tr><tr><td>people_id</td><td>int(11)</td></tr><tr><td>people_name</td><td>varchar(100)</td></tr><tr><td>people_link</td><td>varchar(200)</td></tr><tr><td>people_status</td><td>varchar(100)</td></tr><tr><td>people_title</td><td>varchar(100)</td></tr><tr><td>file_name</td><td>varchar(100)</td></tr></table>	bavardes_site people		people_id	int(11)	people_name	varchar(100)	people_link	varchar(200)	people_status	varchar(100)	people_title	varchar(100)	file_name	varchar(100)						
bavardes_site podcast																															
podcast_id	int(11)																														
podcast_link	varchar(300)																														
podcast_name	varchar(100)																														
file_name	varchar(25)																														
bavardes_site people																															
people_id	int(11)																														
people_name	varchar(100)																														
people_link	varchar(200)																														
people_status	varchar(100)																														
people_title	varchar(100)																														
file_name	varchar(100)																														
<table><tr><th colspan="2">bavardes_site team</th></tr><tr><td>team_id</td><td>int(11)</td></tr><tr><td>team_name</td><td>varchar(50)</td></tr><tr><td>team_title</td><td>varchar(100)</td></tr><tr><td>team_description</td><td>varchar(2000)</td></tr><tr><td>file_name</td><td>varchar(100)</td></tr></table>	bavardes_site team		team_id	int(11)	team_name	varchar(50)	team_title	varchar(100)	team_description	varchar(2000)	file_name	varchar(100)	<table><tr><th colspan="2">bavardes_site users</th></tr><tr><td>us_id</td><td>int(11)</td></tr><tr><td>us_fname</td><td>varchar(25)</td></tr><tr><td>us_lname</td><td>varchar(25)</td></tr><tr><td>us_email</td><td>varchar(50)</td></tr><tr><td>us_password</td><td>varchar(200)</td></tr><tr><td>us_phone</td><td>varchar(10)</td></tr><tr><td>us_status</td><td>tinyint(1)</td></tr><tr><td>file_name</td><td>varchar(25)</td></tr></table>	bavardes_site users		us_id	int(11)	us_fname	varchar(25)	us_lname	varchar(25)	us_email	varchar(50)	us_password	varchar(200)	us_phone	varchar(10)	us_status	tinyint(1)	file_name	varchar(25)
bavardes_site team																															
team_id	int(11)																														
team_name	varchar(50)																														
team_title	varchar(100)																														
team_description	varchar(2000)																														
file_name	varchar(100)																														
bavardes_site users																															
us_id	int(11)																														
us_fname	varchar(25)																														
us_lname	varchar(25)																														
us_email	varchar(50)																														
us_password	varchar(200)																														
us_phone	varchar(10)																														
us_status	tinyint(1)																														
file_name	varchar(25)																														
<table><tr><th colspan="2">bavardes_site collab</th></tr><tr><td>collab_id</td><td>int(11)</td></tr><tr><td>collab_name</td><td>varchar(200)</td></tr><tr><td>collab_link</td><td>varchar(200)</td></tr><tr><td>file_name</td><td>varchar(200)</td></tr><tr><td>collab_cat</td><td>varchar(50)</td></tr></table>	bavardes_site collab		collab_id	int(11)	collab_name	varchar(200)	collab_link	varchar(200)	file_name	varchar(200)	collab_cat	varchar(50)	<table><tr><th colspan="2">bavardes_site presse</th></tr><tr><td>presse_id</td><td>int(11)</td></tr><tr><td>presse_name</td><td>varchar(500)</td></tr><tr><td>presse_link</td><td>varchar(200)</td></tr><tr><td>presse_cat</td><td>varchar(100)</td></tr><tr><td>presse_description</td><td>varchar(500)</td></tr><tr><td>file_name</td><td>varchar(25)</td></tr></table>	bavardes_site presse		presse_id	int(11)	presse_name	varchar(500)	presse_link	varchar(200)	presse_cat	varchar(100)	presse_description	varchar(500)	file_name	varchar(25)				
bavardes_site collab																															
collab_id	int(11)																														
collab_name	varchar(200)																														
collab_link	varchar(200)																														
file_name	varchar(200)																														
collab_cat	varchar(50)																														
bavardes_site presse																															
presse_id	int(11)																														
presse_name	varchar(500)																														
presse_link	varchar(200)																														
presse_cat	varchar(100)																														
presse_description	varchar(500)																														
file_name	varchar(25)																														
<table><tr><th colspan="2">bavardes_site events</th></tr><tr><td>event_id</td><td>int(11)</td></tr><tr><td>event_name</td><td>varchar(100)</td></tr><tr><td>event_date</td><td>date</td></tr><tr><td>event_heure</td><td>varchar(5)</td></tr><tr><td>event_lieu</td><td>varchar(200)</td></tr><tr><td>event_adresse</td><td>varchar(200)</td></tr><tr><td>event_link</td><td>varchar(200)</td></tr><tr><td>event_nombre_pers</td><td>int(4)</td></tr><tr><td>event_inscription</td><td>varchar(50)</td></tr><tr><td>event_non_mix</td><td>varchar(50)</td></tr><tr><td>event_description</td><td>varchar(1000)</td></tr><tr><td>event_cat</td><td>varchar(50)</td></tr><tr><td>file_name</td><td>varchar(50)</td></tr></table>	bavardes_site events		event_id	int(11)	event_name	varchar(100)	event_date	date	event_heure	varchar(5)	event_lieu	varchar(200)	event_adresse	varchar(200)	event_link	varchar(200)	event_nombre_pers	int(4)	event_inscription	varchar(50)	event_non_mix	varchar(50)	event_description	varchar(1000)	event_cat	varchar(50)	file_name	varchar(50)			
bavardes_site events																															
event_id	int(11)																														
event_name	varchar(100)																														
event_date	date																														
event_heure	varchar(5)																														
event_lieu	varchar(200)																														
event_adresse	varchar(200)																														
event_link	varchar(200)																														
event_nombre_pers	int(4)																														
event_inscription	varchar(50)																														
event_non_mix	varchar(50)																														
event_description	varchar(1000)																														
event_cat	varchar(50)																														
file_name	varchar(50)																														

J'ai modifié le fichier database.php pour que la base de données soit liée au projet.

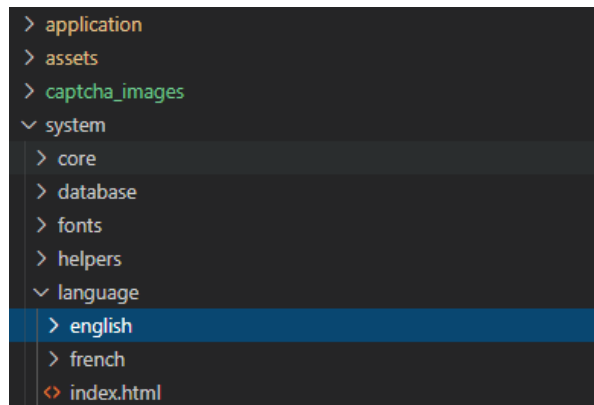
```
application > config > database.php
71  | the query builder class.
72  */
73  $active_group = 'default';
74  $query_builder = TRUE;
75
76  $db['default'] = array(
77      'dsn' => '',
78      'hostname' => 'localhost',
79      'username' => 'root',
80      'password' => '',
81      'database' => 'bavardes_site',
82      'dbdriver' => 'mysqli',
83      'dbprefix' => '',
84      'pconnect' => FALSE,
85      'db_debug' => (ENVIRONMENT !== 'production'),
86      'cache_on' => FALSE,
87      'cachedir' => '',
88      'char_set' => 'utf8',
89      'dbcollat' => 'utf8_general_ci',
90      'swap_pre' => '',
91      'encrypt' => FALSE,
92      'compress' => FALSE,
93      'stricton' => FALSE,
94      'failover' => array(),
95      'save_queries' => TRUE
96  );
97
```

\*Pour la sous-partie 3, je prendrai l'exemple du contrôleur Accueil et de sa vue.

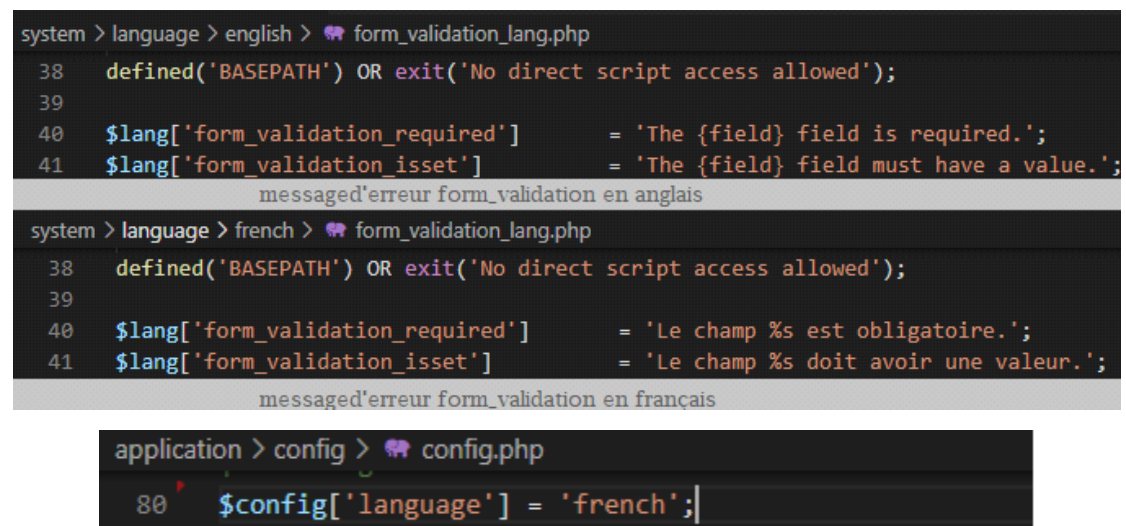
### 3. Version multilingue

La prise en charge multilingue ou l'internationalisation est une caractéristique importante des applications Web.

#### Première étape : La langue du site par défaut

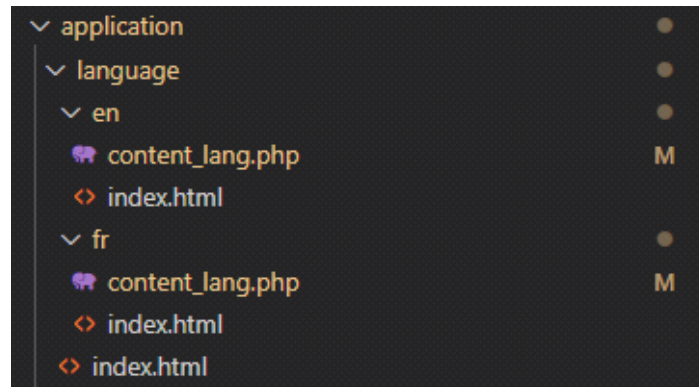


Le langage par défaut de CodeIgniter étant l'anglais, la première chose a été de créer un dossier « french » dans le répertoire « language » de system, dans lequel j'ai reporté tous les fichiers du dossier « english » et les ai traduits en français.



Par la suite j'ai modifié le fichier config.php, situé dans application/config, pour y spécifier la langue du site par défaut.

## Deuxième étape : Création des fichiers de langue



Dans le répertoire « language » de application j'ai créé les dossiers de langue « en » et « fr ».

```
application > language > en > content_lang.php
<?php

defined('BASEPATH') or exit('No direct script access allowed');

//definition de la langue
$lang['lang'] = 'en';

// accueil/index
$lang['acc_button_about'] = $lang['header_about'];
$lang['acc_body_text1'] = ' Les Bavardes is a lesbian, bi, trans, queer and straight feminist collective
committed to the visibility of all women, through cultural and artistic spaces and actions, open to all, aiming
to act, meet, wonder and party. Radio, workshop, debate, cinema, parties... Let's meet, question and emancipate
ourselves! ';
$lang['acc_button_team'] = 'Who Are these Women?';

application > language > fr > content_lang.php
<?php

defined('BASEPATH') or exit('No direct script access allowed');

//définition de la langue
$lang['lang'] = 'fr';

// accueil/index
$lang['acc_button_about'] = $lang['header_about'];
$lang['acc_body_text1'] = ' Les bavardes, c'est un collectif lesbien, bi, trans, queer et hétéro féministe
engagé pour la visibilité de toutes les femmes, au travers d'espaces et d'actions culturelles et artistiques,
ouvert à toutes et tous, visant à agir, se rencontrer, s'interroger et faire la fête. Radio, atelier, débat,
cinéma, soirées... Retrouvons-nous, interrogeons-nous et émancipons-nous ! ';
$lang['acc_button_team'] = 'Qui sont-elles?';
```

Chacun de ces dossiers contient le fichier content\_lang.php avec la définition de la langue (du même nom que son dossier), pour le dossier français tout le contenu est en français, et pour l'anglais j'ai eu à traduire tout le contenu du site en anglais.

## Controller

```
ion > controllers > Accueil.php
<?php

defined('BASEPATH') or exit('No direct script access allowed');

class Accueil extends CI_Controller{
    public function index($lang = ''){

        $data = array();
        $data['controller'] = 'accueil';
        $data['function'] = 'index';
        $this->lang->load('content', $lang == ''?'fr':$lang);
        //définition de la langue
        $data['lang'] = $this->lang->line('lang');

        //accueil/index
        $data['acc_button_about'] = $this->lang->line('acc_button_about');
        $data['acc_body_text1'] = $this->lang->line('acc_body_text1');
        $data['acc_button_team'] = $this->lang->line('acc_button_team');

        $data['acc_body_text2'] = $this->lang->line('acc_body_text2');
        $data['acc_button_pod'] = $this->lang->line('acc_button_pod');

        $data['acc_titre2'] = $this->lang->line('acc_titre2');
        $data['acc_titre2_p1'] = $this->lang->line('acc_titre2_p1');
        $data['acc_titre2_button_p2'] = $this->lang->line('acc_titre2_button_p2');

        $data['acc_titre3'] = $this->lang->line('acc_titre3');
        $data['acc_titre3_button_p1'] = $this->lang->line('acc_titre3_button_p1');

        $this->load->view('layouts/header', $data);
        $this->load->view('accueil/index', $data);
        $this->load->view('layouts/footer', $data);

    } // index ends here
} // controllers ends here
?>
```

La méthode **index()** a en paramètre `$lang = ''`, `$lang` est initialisé.

`$data` est le tableau qui contiendra toutes les données à transmettre à la vue.

Le fichier content est chargé, la langue définit dans la variable `$lang` s'applique sur ce fichier. La valeur de langue est définit par l'opérateur ternaire « `$lang == '' ? 'fr' : $lang` », qui signifie que si `$lang` n'a pas de valeur, sa valeur sera le français, sinon sa valeur sera celle de la langue sélectionnée.

Définition de la langue appelée dans le répertoire language.

Est appelé le contenu du dossier language, que l'on place dans une variable `$data`.

La vue est chargée avec `$data` en 2ème argument afin de faire apparaître les variables dans la vue.

## View

```
application > views > accueil > index.php

?>" role="button"><div class="text-center"><?php echo $acc_button_about; ?></div></a>
?> echo $acc_body_text1; ?>"

" role="button"><div class="text-center">&#10150;<?php echo $acc_button_team; ?></div></a>
```



Dans la vue j'ai transféré les variables du contrôleur.

```
<div class="float-right">
  <a href="{?php echo base_url().'.index.php/'.$controller.'/'.'.function.'/'fr'; }"></a>
  <a href="{?php echo base_url().'.index.php/'.$controller.'/'.'.function.'/'en'; }"></a>
</div>
```

`base_url()` vaut le nom du domaine, `$controller` prend la valeur de la page sur laquelle on se trouve, `$function` a la valeur de la méthode sur laquelle agit la version multilingue et à la fin « en » ou « fr » en fonction du drapeau.



En appuyant l'image du drapeau français le contenu de la page sera traduit en français, en appuyant sur le drapeau britannique le contenu de la page sera traduit en anglais.



\*Pour les sous-parties 4 et 5, je prendrai l'exemple du contrôleur Users, de ses vues, et du modèle User\_Model.

#### 4. Création de session

Le système d'authentification est indispensable afin de restreindre l'accès à notre espace gestion. Afin d'enregistrer les informations de l'utilisateur, il est nécessaire de créer une table, ici j'ai créé la table users :

```
CREATE TABLE IF NOT EXISTS `users` (
  `us_id` int(11) NOT NULL AUTO_INCREMENT,
  `us_fname` varchar(25) NOT NULL,
  `us_lname` varchar(25) NOT NULL,
  `us_email` varchar(50) NOT NULL,
  `us_password` varchar(200) NOT NULL,
  `us_phone` varchar(10) DEFAULT NULL,
  `us_status` tinyint(1) NOT NULL DEFAULT 0 COMMENT '1=Admin | 0=User',
  `file_name` varchar(25) DEFAULT NULL,
  PRIMARY KEY (`us_id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8;
```

Seigneur: MariaDB:3306 » Base de données: bavardes_site » Table: users									
Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Déclencheurs									
Structure de table Vue relationnelle									
#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	us_id	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/> 2	us_fname	varchar(25)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 3	us_lname	varchar(25)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 4	us_email	varchar(50)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 5	us_password	varchar(200)	utf8_general_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 6	us_phone	varchar(10)	utf8_general_ci		Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 7	us_status	tinyint(1)			Non	0	1=Admin   0=User		Modifier Supprimer Plus
<input type="checkbox"/> 8	file_name	varchar(25)	utf8_general_ci		Oui	NULL			Modifier Supprimer Plus

## Model

Le modèle User\_Model gère les opérations liées à la base de données (extraction et insertion).

```
ation > models > User_Model.php
<?php defined('BASEPATH') OR exit('No direct script access allowed');

class User_Model extends CI_Model{
    function __construct() {
        // Set table name
        $this->table = 'users';
    }
    function getRowsUser($params = array()){
        $this->db->select('*');
        $this->db->from($this->table);

        if(array_key_exists("conditions", $params)){
            foreach($params['conditions'] as $key => $val){
                $this->db->where($key, $val);
            }
        }

        if(array_key_exists("returnType",$params) && $params['returnType'] == 'count'){
            $result = $this->db->count_all_results();
        }else{
            if(array_key_exists("us_id", $params) || $params['returnType'] == 'single'){
                if(!empty($params['us_id'])){
                    $this->db->where('us_id', $params['us_id']);
                }
                $query = $this->db->get();
                $result = $query->row_array();
            }else{
                if(array_key_exists("start",$params) && array_key_exists("limit",$params)){
                    $this->db->limit($params['limit'],$params['start']);
                }elseif(!array_key_exists("start",$params) && array_key_exists("limit",$params)){
                    $this->db->limit($params['limit']);
                }

                $query = $this->db->get();
                $result = ($query->num_rows() > 0)?$query->result_array():FALSE;
            }
        }

        // Return fetched data
        return $result;
    } // getRowsUsers
    public function insert($data = array()) {
        if(!empty($data)){
            // Insert member data
            $insert = $this->db->insert($this->table, $data);

            // Return the status
            return $insert?true:false;
        }
        return false;
    } // insert ends here
}
```

La méthode **\_\_construct()** permet de pouvoir utiliser les éléments qu'on place dans cette fonction sur toute la classe. De cette manière, je n'ai pas besoin de charger modèle, langage, paramètres dans toutes les méthodes de la classe.

La méthode **getRowsUser()** permet de récupérer les données d'utilisateur de la base de données en fonction des conditions.

La méthode **insert()** insère les données du compte utilisateur dans la base de données.

## Controller

Le contrôleur Users gère les opérations d'enregistrement et d'authentification liées à l'utilisateur.

```
ion > controllers > Users.php

class Users extends CI_Controller {

    function __construct() {
        parent::__construct();

        // Load form validation library & user model
        $this->load->helper('form');
        $this->load->helper('captcha');
        $this->load->library('form_validation');

        $this->load->model('User_Model');

        $this->load->library('pagination');
        $this->perPage = 10;

        // File upload path
        $this->uploadPath = 'uploads/admin_img/';

        // User login status
        $this->isUserLoggedIn = $this->session->userdata('isUserLoggedIn');
    } // __construct ends here
}
```

La méthode **\_\_construct()** permet de charger bibliothèque, assistants, le modèle requis et le statut de la connexion de l'utilisateur dans l'objet `$this->isUserLoggedIn`.

```

public function admin(){

    $data['controller'] = 'users';
    $data['function'] = 'admin';

    if($this->isUserLoggedIn){
        redirect('users/ad_index');
    }else{
        redirect('users/login');
    }
} // admin ends here

```

La méthode **admin()** redirige l'utilisateur soit, vers la page de l'espace admin soit, vers la page de connexion en fonction du statut de connexion.

```

643 public function account($lang = ''){
644     $data = array();
645
646     $this->lang->load('content', $lang == ''?'fr':$lang);
647     $data['controller'] = 'users';
648     $data['function'] = 'account';
649
650     if($this->isUserLoggedIn){
651         $con = array(
652             'us_id' => $this->session->userdata('userId')
653         );
654         $data['ma_pages'] = 'account';
655         $data['user'] = $this->User_Model->getRowsUser($con);
656
657         //définition de la langue
658         $data['lang'] = $this->lang->line('lang');
659
660
661         code raccourcis, partie $lang
        coupé
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708     $data['controller'] = 'users';
709     $data['ma_pages'] = $data['function'] = 'account';
710
711     // Pass the user data and load view
712     $this->load->view('layouts/header', $data);
713     $this->load->view('users/account', $data);
714     $this->load->view('layouts/footer', $data);
715 }else{
716     redirect('users/login');
717 }
718 } // account ends here

```

La méthode **account()** obtient les informations de compte de l'utilisateur connecté à l'aide de la méthode `getRowsUser()` du modèle et transmet les données de l'utilisateur.

`$data` est le tableau qui contiendra toutes les données à transmettre à la vue. On charge la vue du compte de l'utilisateur avec `$data` en 2ème argument afin de faire apparaître les variables dans la vue.

```
// If registration request is submitted
if($this->input->post('signupSubmit')){
    $this->form_validation->set_rules('us_fname', $data['create_fval_fname'] = $this->lang->line('create_fval_fname'), 'required|alpha');
    $this->form_validation->set_rules('us_lname', $data['create_fval_lname'] = $this->lang->line('create_fval_lname'), 'required|alpha');
    $this->form_validation->set_rules('us_email', $data['create_fval_email'] = $this->lang->line('create_fval_email'),
    'required|valid_email|callback_email_check');
    $this->form_validation->set_rules('us_phone', $data['create_fval_phone'] = $this->lang->line('create_fval_phone'), 'required|integer');
    $this->form_validation->set_rules('us_password', $data['create_fval_psw'] = $this->lang->line('create_fval_psw'), 'required');
    $this->form_validation->set_rules(
        'us_conf_password', $data['create_fval_conf_psw'] = $this->lang->line('create_fval_conf_psw'), 'required|matches[us_password]'
    );
    $this->form_validation->set_rules(
        'user_captcha', $data['create_fval_captcha'] = $this->lang->line('create_fval_captcha'), 'required|callback_check_captcha'
    );
    $user_captcha = $this->input->post('user_captcha');

    $userData = array(
        'us_fname' => strip_tags($this->input->post('us_fname',true)),
        'us_lname' => strip_tags($this->input->post('us_lname',true)),
        'us_email' => strip_tags($this->input->post('us_email',true)),
        'us_password' => sha1($this->input->post('us_password',true)),
        'us_phone' => strip_tags($this->input->post('us_phone',true)),
    );

    if($this->form_validation->run() == true){
        $insert = $this->User_Model->insert($userData);
        if($insert){
            $this->session->set_userdata('success_msg', $this->lang->line('create_suc_msg'));
            redirect('users/login/'.$lang);
        }else{
            $data['error_msg'] = $this->lang->line('create_err_msg1');
        }
    }else{
        $data['error_msg'] = $this->lang->line('create_err_msg2');
    }
}
}
```

```
if($this->form_validation->run() == false){
    $random_number = substr(number_format(time()*rand(),0,'',0),0,6);

    $vals = array(
        'word' => $random_number,
        'img_path' => './captcha_images/',
        'img_url' => base_url().'captcha_images/',
        'img_width' => 140,
        'img_height' => 32,
        'expiration' => 7200
    );

    $data['captcha'] = create_captcha($vals);
    $this->session->set_userdata('captchaWord', $data['captcha']['word']);
}
```

La méthode **registration()** charge initialement la vue du formulaire d'inscription avec \$data (le tableau qui contiendra toutes les données à transmettre à la vue) en 2ème argument afin de faire apparaître les variables dans la vue.

Si le formulaire est soumis :

- Elle vérifie les données mises dans les champs de saisie en utilisant la bibliothèque form\_validation.
- Elle vérifie si l'adresse mail insérer n'existe pas déjà dans la base de données en utilisant une fonction de rappel personnalisée (email\_check).
- Elle vérifie, en utilisant une fonction de rappel personnalisée (check\_captcha), si le nombre inséré dans le captcha correspond bien à l'image du captcha.

Dans le cas où formulaire est vérifié et validé, la méthode `registration()` insère les informations de l'utilisateur dans la base de données en utilisant la méthode `insert()` du modèle.

```
// Existing email check during validation
public function email_check($str){
    $con = array(
        'returnType' => 'count',
        'conditions' => array(
            'us_email' => $str
        )
    );
    $checkEmail = $this->User_Model->getRowsUser($con);
    if($checkEmail > 0){
        $this->form_validation->set_message('email_check', $this->lang->line('create_mail_check'));
        return FALSE;
    }else{
        return TRUE;
    }
} // email_check ends here
```

La méthode `email_check()` est la fonction de rappel personnalisée utilisée avec la bibliothèque `form_validation` pour vérifier si l'adresse mail est déjà existante dans la base de données.

```
public function check_captcha($str){
    $word = $this->session->userdata('captchaWord');
    if(strcmp(strtoupper($str), strtoupper($word)) == 0){
        return true;
    } else {
        $this->form_validation->set_message('check_captcha', $data['create_captcha'] = $this->lang->line('create_captcha'));
        return false;
    }
} // check_captcha ends here
```

La méthode `check_captcha()` est la fonction de rappel personnalisée utilisée avec la bibliothèque `form_validation` pour vérifier si le motif écrit dans le champ de saisie du captcha correspond bien à l'image de ce dernier.

```
574 // If login request submitted
575 if($this->input->post('loginSubmit')){
576     $this->form_validation->set_rules('us_email', $data['login_fval_psw'] =
577         $this->lang->line('login_fval_psw'), 'required|valid_email');
578     $this->form_validation->set_rules('us_password', $data['login_fval_email'] =
579         $this->lang->line('login_fval_email'), 'required');
580
581     if($this->form_validation->run() == true){
582         $con = array(
583             'returnType' => 'single',
584             'conditions' => array(
585                 'us_email' => $this->input->post('us_email', true),
586                 'us_password' => sha1($this->input->post('us_password', true)),
587                 'us_status' => 0
588             )
589         );
590     };
591     $checkLogin = $this->User_Model->getRowsUser($con);
592     if($checkLogin){
593         $this->session->set_userdata('isUserLoggedIn', TRUE);
594         $this->session->set_userdata('userId', $checkLogin['us_id']);
595         redirect('users/account/'.$lang);
596     }else{
597         $data['error_msg'] = $this->lang->line('login_err_msg1');
598     }
599 }else{
600     $data['error_msg'] = $this->lang->line('login_err_msg2');
601 }
```



```

601 if($this->form_validation->run() == true){
602     $con = array(
603         'returnType' => 'single',
604         'conditions' => array(
605             'us_email' => $this->input->post('us_email',true),
606             'us_password' => sha1($this->input->post('us_password',true)),
607             'us_status' => 1
608         )
609     );
610     $checkLogin = $this->User_Model->getRowsUser($con);
611     if($checkLogin){
612         $this->session->set_userdata('isUserLoggedIn', TRUE);
613         $this->session->set_userdata('userId', $checkLogin['us_id']);
614         redirect('users/admin');
615     }else{
616         $data['error_msg'] = $this->lang->line('login_err_msg1');
617     }
618 }else{
619     $data['error_msg'] = $this->lang->line('login_err_msg2');
620 }
621 }
622
623
624 $data['controller'] = 'users';
625 $data['ma_pages'] = $data['function'] = 'login';
626
627 // Load view
628 $this->load->view('layouts/header', $data);
629 $this->load->view('users/login', $data);
630 $this->load->view('layouts/footer', $data);
631
632 } // login ends here

```

La méthode **login()** charge initialement la vue du formulaire de connexion avec \$data (le tableau qui contiendra toutes les données à transmettre à la vue) en zème argument afin de faire apparaître les variables dans la vue.

Si le formulaire est soumis :

- Les données mises dans les champs de saisie sont validées en utilisant la bibliothèque form\_validation.
- La méthode vérifie, à l'aide de la variable \$checkLogin, si l'utilisateur existe dans la base de données avec les informations de connexion fournies.

Dans le cas où formulaire est vérifié et validé, la méthode va faire une dernière vérification. Elle va vérifier le numéro présent dans la colonne « us\_status ». Si il y a le chiffre « 0 », cela signifie que la personne qui se connecte est un utilisateur lambda, elle sera donc dirigée vers sa page de compte utilisateur, mais si la valeur est de « 1 », cela signifie que la personne qui se connecte est un admin, elle sera donc dirigée vers l'espace admin.

```

public function logout($lang = ''){
    $this->lang->load('content', $lang == '?'?'fr':$lang);
    $this->session->unset_userdata('isUserLoggedIn');
    $this->session->unset_userdata('userId');
    $this->session->sess_destroy();
    redirect('users/login/'.$lang);
} //logout ends here

```

La méthode **logout()** permet de pouvoir déconnecter l'utilisateur de son compte en écrasant sa session.

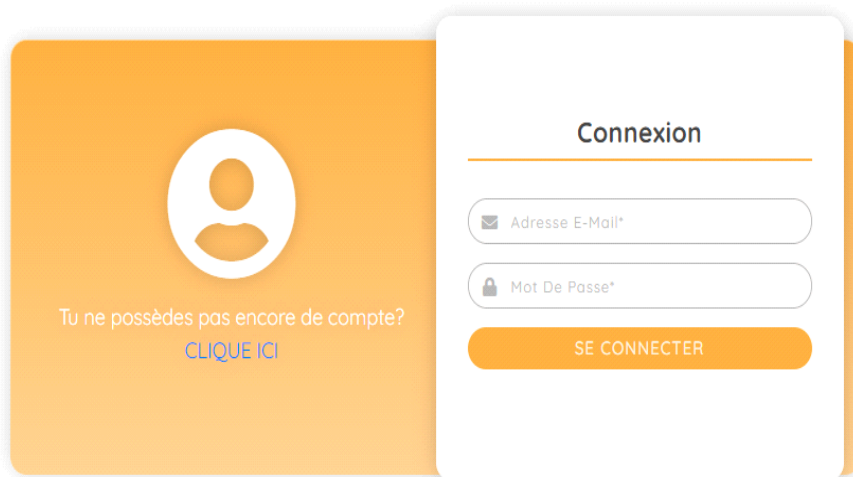
## View

Dans les vues suivantes j'ai transféré les variables du contrôleur.

```
<div class="row">
  <div class="col-6">
    <div class="form-group">
      <label for="us_password"><?php echo $create_psw; ?>* :</label>
      <input type="password" name="us_password" id="us_password" class="form-control" required>
      <span class="missPsw" id="missPsw"></span>
      <?php echo form_error('us_password','<p class="help-block text-danger">','</p>'); ?>
    </div>
  </div>
  <div class="col-6">
    <div class="form-group">
      <label for="us_conf_password"><?php echo $create_psw_cf; ?>* :</label>
      <input type="password" class="form-control" id="us_conf_password" name="us_conf_password" required>
      <span class="missConfPsw" id="missConfPsw"></span>
      <?php echo form_error('us_conf_password','<p class="help-block text-danger">','</p>'); ?>
    </div>
  </div>
</div>
<fieldset>
<legend><?php echo $create_captcha_legend; ?></legend>
  <div class="row">
    <div class="col-5">
      <div class="form-group">
        <input type="text" name="user_captcha" id="user_captcha" placeholder="<?php echo $create_captcha_text; ?>*"
        class="form-control" value="<?php if(!empty($user_captcha)){echo $user_captcha;} ?>" required>
        <span class="missCaptcha" id="missCaptcha"></span>
        <?php echo form_error('user_captcha','<p class="help-block text-danger">','</p>'); ?>
      </div>
    </div>
    <div class="col-5">
      <div class="form-group">
        <span><?php echo $captcha['image']; ?></span>
      </div>
    </div>
  </div>
</fieldset>
<div class="form-group">
  <input type="reset" class="btn btn-secondary" value="<?php echo $create_reset; ?>">
  <input type="submit" name="signupSubmit" class="btn btn-primary" value="<?php echo $create_submit; ?>">
</div>
```

Le fichier *registration.php* contient le formulaire d'inscription. Une fois que l'utilisateur a rempli et validé le formulaire, il est soumis à la méthode *registration()* du contrôleur.





```
<form action="" method="post">
  <div class="form-group">
    <label for="us_email"><?php echo $login_mail; ?> :</label>
    <input type="email" name="us_email" class="form-control" >
    <?php echo form_error('us_email','<p class="help-block text-danger">','</p>'); ?>
  </div>
  <div class="form-group">
    <label for="us_password"><?php echo $login_psw; ?> :</label>
    <input type="password" name="us_password" class="form-control">
    <?php echo form_error('us_password','<p class="help-block text-danger">','</p>'); ?>
  </div>
  <div class="send-button">
    <input type="reset" class="btn btn-secondary" value="<?php echo $login_reset; ?>">
    <input type="submit" name="loginSubmit" class="btn btn-primary" value="<?php echo $login_submit; ?>">
  </div>
</form>
<p><?php echo $login_act; ?> <a href="<?php echo base_url(). 'Users/registration/' . $lang; ?>"><?php echo $login_clk; ?></a></p>
```

Le fichier *login.php* contient le formulaire de connexion. Une fois que l'utilisateur a rempli et validé le formulaire, il est soumis à la méthode *login()* du contrôleur.

```
1 <div class="container">
2   <a class="mb-4" href="<?php echo base_url(). 'users/logout/' . $lang; ?>" class="float-right"><?php echo $account_logout; ?></a>
3   <h3 class="m-4">
4     <p class="text-center">
5       <?php echo $account_wlc; ?> <?php echo $user['us_fname']; ?> <?php echo $account_wlc_p2; ?>.
6     </p>
7     <p class="text-center">
8       <?php echo $account_btn_event; ?> <a href="<?php echo base_url(). 'event/index'; ?>"
9       class="btn btn-dark"><?php echo $account_btn_event_ici; ?> <i class="fa fa-calendar" aria-hidden="true"></i></a>
10    </p>
11    <p class="text-center">
12      <?php echo $account_asso; ?>
13      <a href="https://www.helloasso.com/associations/les-bavardes/adhesions/adhesion-2020" class="link" target="_blank">
14        <button class="button2 btn-block mt-3"><?php echo $account_asso_ici; ?></button></a>
15    </p>
16  </h3>
17  <h4><?php echo $account_info ?> : </h4>
18  <div class="mb-5">
19    <p class="mb-4"><b><?php echo $account_name; ?> : </b><?php echo $user['us_fname']. ' '. $user['us_lname']; ?></p>
20    <p class="mb-4"><b><?php echo $account_mail; ?> : </b><?php echo $user['us_email']; ?></p>
21    <p class="mb-4"><b><?php echo $account_phone; ?> : </b><?php echo $user['us_phone']; ?></p>
22  </div>
23 </div>
```

Le fichier *account.php* affiche les détails du compte de l'utilisateur connecté.

## 5. Réalisation d'un CRUD

Les opérations CRUD sont les fonctionnalités les plus utilisées dans CodeIgniter. Le CRUD permet de manipuler les données dans les bases de données. On peut ajouter, afficher, mettre à jour et supprimer des données de la base de données avec le CRUD. Dans cette partie le CRUD permet de pouvoir gérer les utilisateurs admin, la création d'un nouvel admin, de modifier ses informations personnelles ou bien de supprimer le membre.

Pour cela il faut une base de données et une table. Nous allons donc reprendre la base de données du site ainsi que la table « users » (sa structure se situe dans la partie sur les sessions).

### Model

```
class User_Model extends CI_Model{
    function __construct() {
        // Set table name
        $this->table = 'users';
    }

    * Returns rows from the database based on the conditions
    * @param array filter data based on the passed parameters
    */
    public function getRowsAdmin($params = array()){
        $this->db->select('*');
        $this->db->from($this->table);

        if(array_key_exists("conditions", $params)){
            foreach($params['conditions'] as $key => $val){
                $this->db->where($key, $val);
            }
        }

        if(!empty($params['searchKeywordAdmin'])){
            $search = $params['searchKeywordAdmin'];
            $likeArr = array('us_fname' => $search);
            $this->db->or_like($likeArr);
        }

        if(array_key_exists('returnType', $params) && $params['returnType'] == 'count'){
            $result = $this->db->count_all_results();
        } else {
            if(array_key_exists("us_id", $params)){
                $this->db->where('us_id', $params['us_id']);
                $query = $this->db->get();
                $result = $query->row_array();
            }else{
                if(array_key_exists("start", $params) && array_key_exists("limit", $params)){
                    $this->db->limit($params['limit'], $params['start']);
                }elseif(!array_key_exists("start", $params) && array_key_exists("limit", $params)){
                    $this->db->limit($params['limit']);
                }

                $query = $this->db->get();
                $result = ($query->num_rows() > 0)?$query->result_array():FALSE;
            }
        }

        // Return fetched data
        return $result;
    } //getRowsAdmin ends here
}
```

```

    @param $data data to be inserted
    */
    public function insert($data = array()) {
        if(!empty($data)){

            // Insert member data
            $insert = $this->db->insert($this->table, $data);

            // Return the status
            return $insert?true:false;
        }
        return false;
    } // insert ends here


    /**
     * Update users data into the database
     * @param $data array to be update based on the passed parameters
     * @param $id num filter data
     */
    public function update($data, $id) {
        if(!empty($data) && !empty($id)){

            // Update users data
            $update = $this->db->update($this->table, $data, array('us_id' => $id));

            // Return the status
            return $update?true:false;
        }

    } //update ends here


    public function delete($id){
        // Delete users data
        $delete = $this->db->delete($this->table, array('us_id' => $id));

        // Return the status
        return $delete?true:false;
    } //delete ends here

```

Le modèle User\_Model contient les méthodes suivantes pour manipuler les opérations liées à la base de données.

La méthode **\_\_construct()** a le nom de la table de définit (« users » comme dans la sous partie ci-dessus).

La méthode **getRowsAdmin()** récupère les données des membres de la base de données en fonction des paramètres spécifiés passés dans les \$params. Elle renvoie également les enregistrements filtrés en cas de succès.

La méthode **insert()** insère les données des membres dans la base de données. Elle permet aussi de renvoyer l'ID de ligne en cas de succès et FALSE en cas d'erreur.

La méthode **update()** met à jour les données des membres admin dans la base de données en fonction de l'ID de ligne. Elle retourne TRUE en cas de succès et FALSE en cas d'erreur.

La méthode **delete()** supprime le membre admin de la base de données en fonction de l'ID de ligne. Elle retourne TRUE en cas de succès et FALSE en cas d'erreur.

## Controller

Le contrôleur Users gère les opérations CRUD avec les méthodes suivantes :

```
tion > controllers > Users.php

class Users extends CI_Controller {

    function __construct() {
        parent::__construct();

        // Load form validation library & user model
        $this->load->helper('form');
        $this->load->helper('captcha');
        $this->load->library('form_validation');

        $this->load->model('User_Model');

        $this->load->library('pagination');
        $this->perPage = 10;

        // File upload path
        $this->uploadPath = 'uploads/admin_img/';

        // User login status

        $this->isUserLoggedIn = $this->session->userdata('isUserLoggedIn');
    } // __construct ends here
}
```

La méthode **\_\_construct()** permet de charger bibliothèque, assistants, le modèle requis et le chemin du fichier téléchargé dans l'objet \$this->uploadPath.

```

46 public function ad_index(){
47
48     if($this->isUserLoggedIn){
49         $data = array();
50
51         // Get messages from the session
52         if($this->session->userdata('success_msg')){
53             $data['success_msg'] = $this->session->userdata('success_msg');
54             $this->session->unset_userdata('success_msg');
55         }
56         if($this->session->userdata('error_msg')){
57             $data['error_msg'] = $this->session->userdata('error_msg');
58             $this->session->unset_userdata('error_msg');
59         }
60
61         //If search request is submitted
62         if($this->input->post('submitSearch')){
63             $inputKeywords = $this->input->post('searchKeywordAdmin');
64             $searchKeywordAdmin = strip_tags($inputKeywords);
65             if(!empty($searchKeywordAdmin)){
66                 $this->session->set_userdata('searchKeywordAdmin', $searchKeywordAdmin);
67             } else {
68                 $this->session->unset_userdata('searchKeywordAdmin');
69             }
70         } elseif($this->input->post('submitSearchReset')){
71             $this->session->unset_userdata('searchKeywordAdmin');
72         }
73         $data['searchKeywordAdmin'] = $this->session->userdata('searchKeywordAdmin');
74
75         //Get Rows count
76         $conditions['searchKeywordAdmin'] = $data['searchKeywordAdmin'];
77         $conditions['returnType'] = 'count';
78         $rowsCount = $this->User_Model->getRowsAdmin($conditions);
79
80         //Pagination config
81         $config['base_url'] = base_url(). 'index.php/users/ad_index/';
82         $config['uri_segment'] = 3;
83         $config['total_rows'] = $rowsCount;
84         $config['per_page'] = $this->perPage;
85
86         //Initialize pagination library
87         $this->pagination->initialize($config);
88
89         //Define offset
90         $page = $this->uri->segment(3);
91         $offset = !$page?0:$page;
92
93         //Get rows
94         $conditions['returnType'] = '';
95         $conditions['start'] = $offset;
96         $conditions['limit'] = $this->perPage;
97         $data['admin'] = $this->User_Model->getRowsAdmin($conditions);
98
99         $data['ma_pages'] = 'ad_index_admin';
100         $data['controller'] = 'users';
101         $data['function'] = 'ad_index';
102
103         // Load the list page view
104         $this->load->view('layouts/adheader', $data);
105         $this->load->view('ad_admin/ad_index', $data);
106         $this->load->view('layouts/adfooter', $data);
107
108     } else {
109         redirect('users/login');
110     }
111 } // ad_index ends here

```



La méthode ***ad\_index()*** liste les données des utilisateurs admin en utilisant la méthode ***getRowsAdmin()*** du modèle, transmet les données des membres admin et charge la vue avec ***\$data*** (le tableau qui contiendra toutes les données à transmettre à la vue) en 2ème argument afin de faire apparaître les variables dans la vue.

```
// If create request is submitted
if($this->input->post('admin_add')){
    // Form field validation rules
    $this->form_validation->set_rules('us_lname','nom','required');
    $this->form_validation->set_rules('us_fname','prénom','required');
    $this->form_validation->set_rules('us_password','mot de passe','required');
    $this->form_validation->set_rules('us_phone','n° de téléphone','integer');
    $this->form_validation->set_rules('us_email','adresse e-mail','required|valid_email');
    $this->form_validation->set_rules('us_status','statut admin','required');

    // Prepare gallery data
    $formArray = array(
        'us_fname' => $this->input->post('us_fname',TRUE),
        'us_lname' => $this->input->post('us_lname',TRUE),
        'us_password' => sha1($this->input->post('us_password',TRUE)),
        'us_phone' => $this->input->post('us_phone',TRUE),
        'us_email' => $this->input->post('us_email',TRUE),
        'us_status' => $this->input->post('us_status',TRUE)
    );
}
```

La méthode ***add\_admin()*** ajoute les données du membre admin à la base de données en utilisant la méthode ***insert()*** du modèle.

```
// Validate submitted form data
if($this->form_validation->run() == true){
    // Upload image file to the server
    if(!empty($_FILES['image']['name'])){
        $imageName = $_FILES['image']['name'];

        // File upload configuration
        $config['upload_path'] = $this->uploadPath;
        $config['allowed_types'] = 'jpg|jpeg|png|gif';

        // Load and initialize upload library
        $this->load->library('upload', $config);
        $this->upload->initialize($config);

        // Upload file to server
        if($this->upload->do_upload('image')){
            // Uploaded file data
            $fileData = $this->upload->data();
            $formArray['file_name'] = $fileData['file_name'];
        }else{
            $error = $this->upload->display_errors();
        }
    }

    if(empty($error)){
        // Insert data
        $insert = $this->User_Model->insert($formArray);

        if($insert){
            $this->session->set_userdata('success_msg', 'Ajout - Réussie.');
            redirect('users/ad_index');
        }else{
            $error = 'Quelques problèmes sont survenus, veuillez réessayer.';
        }
    }

    $data['error_msg'] = $error;
}
}
```

```

$data['admin'] = $formArray;

$data['controller'] = 'users';
$data['ma_pages'] = $data['function'] = 'add_admin';

// Load the add page view
$this->load->view('layouts/adheader', $data);
$this->load->view('ad_admin/add', $data);
$this->load->view('layouts/adfooter');

```

La vue du formulaire est initialement chargée pour recevoir les entrées de l'utilisateur avec \$data (le tableau qui contiendra toutes les données à transmettre à la vue) en 2ème argument afin de faire apparaître les variables dans la vue.

Si le formulaire est soumis :

- Les données de formulaire publiées sont validées à l'aide de la bibliothèque de form\_validation.
- Insère une image avec la fonctionnalité upload en fonction des conditions de taille, d'extension, l'image est recueillie dans le dossier dont le chemin est placé dans \$this->uploadPath.

Dans le cas où le formulaire d'ajout est validé par la méthode add\_admin(), cette dernière insère les données du nouvel admin dans la base de données à l'aide de la méthode insert() du modèle.

```

public function edit_admin($id){

    if($this->isUserLoggedIn){

        $data = $formArray = array();

        // Get image data
        $con = array('us_id' => $id);
        $formArray = $this->User_Model->getRowsAdmin($con);
        $prevFArray = $formArray['file_name'];

        // If update request is submitted
        if($this->input->post('admin_edit')){
            // Form field validation rules
            $this->form_validation->set_rules('us_lname','nom','required');
            $this->form_validation->set_rules('us_fname','prénom','required');
            $this->form_validation->set_rules('us_password','mot de passe','required');
            $this->form_validation->set_rules('us_phone','n° de téléphone','integer');
            $this->form_validation->set_rules('us_email','adresse e-mail','required');
            $this->form_validation->set_rules('us_status','statut admin','required');

            // Prepare gallery data
            $formArray = array(
                'us_fname' => $this->input->post('us_fname',TRUE),
                'us_lname' => $this->input->post('us_lname',TRUE),
                'us_password' => sha1($this->input->post('us_password',TRUE)),
                'us_phone' => $this->input->post('us_phone',TRUE),
                'us_email' => $this->input->post('us_email',TRUE),
                'us_status' => $this->input->post('us_status',TRUE)
            );
        }
    }
}

```

```

// Validate submitted form data
if($this->form_validation->run() == true){
    // Upload image file to the server
    if(!empty($_FILES['image']['name'])){
        $imageName = $_FILES['image']['name'];

        // File upload configuration
        $config['upload_path'] = $this->uploadPath;
        $config['allowed_types'] = 'jpg|jpeg|png|gif';

        // Load and initialize upload library
        $this->load->library('upload', $config);
        $this->upload->initialize($config);

        // Upload file to server
        if($this->upload->do_upload('image')){
            // Uploaded file data
            $fileData = $this->upload->data();
            $formArray['file_name'] = $fileData['file_name'];

            // Remove old file from the server
            if(!empty($prevFArray)){
                @unlink($this->uploadPath.$prevFArray);
            }
        }else{
            $error = $this->upload->display_errors();
        }
    }

    if(empty($error)){
        // Update image data
        $update = $this->User_Model->update($formArray, $id);

        if($update){
            $this->session->set_userdata('success_msg', 'Mise à jour - Réussie.');
```

redirect('users/ad\_index');

```

        }else{
            $error = 'Quelques problèmes sont survenus, veuillez réessayer.';
        }
    }

    $data['error_msg'] = $error;
}
}
}

```

La méthode ***edit\_admin()*** a \$id en paramètre modifie et met à jour les données du membre spécifique en utilisant la méthode update() du modèle. Elle récupère les données de l'admin de la base de données en fonction de l'ID spécifique. La vue du formulaire est chargée avec les données de l'admin pré remplies et avec \$data (le tableau qui contiendra toutes les données à transmettre à la vue) en 2ème argument afin de faire apparaître les variables dans la vue.

Si le formulaire est soumis :

- Les données de formulaire publiées sont validées à l'aide de la bibliothèque de form\_validation.
- Si l'image est modifiée, la nouvelle image, avec la fonctionnalité upload en fonction des conditions de taille, d'extension, prend la place de l'ancienne, pas seulement sur la page web mais aussi dans le dossier dont le chemin est placé dans \$this->uploadPath. L'ancienne image est écrasée par la nouvelle.

Dans le cas où le formulaire de modification est validé par la méthode edit\_admin(), cette dernière met à jour les données de l'admin dans la base de données à l'aide de la méthode update() du modèle.



```

public function delete_admin($id){

    if($this->isUserLoggedIn){
        // Check whether id is not empty
        if($id){
            $con = array('us_id' => $id);
            $formArray = $this->User_Model->getRowsAdmin($con);

            // Delete users (admin) data
            $delete = $this->User_Model->delete($id);

            if($delete){
                // Remove file from the server
                if(!empty($formArray['file_name'])){
                    @unlink($this->uploadPath.$formArray['file_name']);
                }

                $this->session->set_userdata('success_msg', 'Suppression - Réussie.');
                redirect('users/ad_index');
            }else{
                $this->session->set_userdata('error_msg', 'Veuillez réessayer');
            }
        }
    }

    } else {
        redirect('users/login');
    }

}

} // delete_admin ends here

```

La méthode ***delete\_admin()*** supprime les données du membre admin de la base de données à l'aide de la méthode ***delete()*** du modèle, ainsi que l'image dans le dossier dont le chemin est placé dans ***\$this->uploadPath***.

## View

```

</thead>
<tbody>
    <?php if(!empty($admin)){
        foreach($admin as $row){
            if($row['us_status'] == 1){
                $image = !empty($row['file_name'])?'':'';
            }

            <tr>
                <td><?php echo $row['us_id']; ?></td>
                <td><?php echo $image; ?></td>
                <td><?php echo $row['us_lname']; ?></td>
                <td><?php echo $row['us_fname']; ?></td>
                <td><?php echo $row['us_email']; ?></td>
                <td><?php echo $row['us_phone']; ?></td>
                <td>
                    <a href="<?php echo base_url().'.index.php/users/edit_admin/'.$row['us_id']; ?>" class="btn btn-sm btn-warning">
                    <i class="fas fa-user-edit"></i></a>
                    <a href="<?php echo base_url().'.index.php/users/delete_admin/'.$row['us_id']; ?>" class="btn btn-sm btn-danger"
                    onclick="return confirm('Voulez vous vraiment supprimer cette donnée?')"><i class="fas fa-user-minus"></i></a>
                </td>
            </tr>

        <?php }}} else { ?>

        <tr>
            <td colspan="5">Records not found</td>

```

Toutes les données des utilisateurs admin sont récupérées à partir de la base de données et sont répertoriées dans la page web ***ad\_index*** avec le lien « ajouter », « modifier » et « supprimer ».



En cliquant sur le bouton jaune, l'utilisateur admin est redirigé vers le formulaire de modification pour effectuer l'opération de mise à jour. Cette vue est chargée par la fonction `edit_admin()` du contrôleur.

```
<div class="form-group">
  <label>Images (.jpg .jpeg .png .gif) :</label>
  <input type="file" name="image" class="form-control" value="<?php echo !empty($admin['file_name'])?$admin['file_name']:''; ?>"
  <?php echo form_error('image','<p class="help-block text-danger">','</p>'); ?>
  <?php if(!empty($admin['file_name'])) { ?>
    <div class="img-box">
      
    </div>
  <?php } ?>
</div>


<div class="form-group">
  <label for="us_status">Status: </label> <br>
  <input type="checkbox" name="us_status" id="us_status" value="1" <?php if($admin['us_status'] == 1){echo "checked";}?>> Admin
  <?php echo form_error('us_status','<p class="help-block text-danger">','</p>'); ?>
</div>

<div class="form-group">
  <a href="<?php echo base_url(). 'users/ad_index'; ?>" class="btn btn-sm btn-secondary"><i class="fas fa-backspace"></i></a>
  <input type="hidden" name="us_id" value="<?php echo !empty($admin['us_id'])?$admin['us_id']:''; ?>"
  <input type="submit" name="admin_edit" class="btn btn-sm btn-warning" value="Modifier">
</div>
```

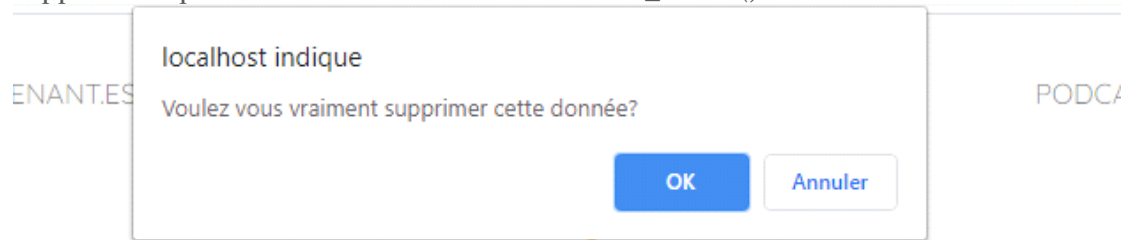
Sur demande de modification, un formulaire s'affiche avec des données de l'admin pré remplis dans les champs de saisie.

LES DYNAMES COLLECTIF FEMINISTE & LESBIEN  
À AMIENS DEPUIS 2017

## MODIFICATION D'UNE ADMIN

Nom :	Prénom :
<input type="text" value="Guehoun"/>	<input type="text" value="Jade"/>
N° Téléphone :	Adresse E-mail :
<input type="text" value="REDACTED"/>	<input type="text" value="REDACTED"/>
Mot de Passe :	
<input type="password" value="REDACTED"/>	
Images (.jpg .jpeg .png .gif) :	
<input type="button" value="Choisir un fichier"/> <input type="text" value="Aucun fichier choisi"/>	
	
Status:	
<input checked="" type="checkbox"/> Admin	
<input type="button" value="Modifier"/>	

En cliquant sur le bouton rouge, apparaît un message de confirmation de suppression du membre admin. Si le bouton "OK" est cliqué, alors est enclenché l'action de suppression que l'on trouve dans la fonction *delete\_admin()* du contrôleur.



## LES BAVARDES COLLECTIF FÉMINISTE & LESBIEN À AMIENS DEPUIS 2017

Cette espace permet à l'administrateur de pouvoir gérer les données de la base de données à même cette espace site. Ici il pourra gérer informations liées aux évènements, aux artistes, aux intervenant.es, aux membres du collectif, aux articles de presse, aux collaborations, aux podcasts et aux membres admins.

### E. Mise en ligne

La mise en ligne du site s'est faite le jeudi 18 juin avec OVH et WinSCP, j'avais déjà mis ce site en ligne sur dev.amorce.org pour le tester en dehors du local. De cette manière j'ai pu rapidement repérer les configurations à modifier. Il a fallu modifier la base\_url par "https://lesbavardes.org/". Ouvrir à partir de l'interface OVH phpmyadmin, y exporter la base de données qui était en local. Et enfin aller dans database.php pour modifier le nom du serveur, de l'utilisateur, de la base de données et le mot de passe.

# Projets réalisés – femmes-citoyennes.fr

## A - Présentation du collectif

Citoyennes Amiénoises Féministes est un collectif de Femmes citoyennes engagé dans la visibilité de toutes les femmes dans le cadre de la campagne municipale Amiénoise 2020.

Leur action est le fruit d'un travail mené par un collectif féministe non mixte, portant les revendications des femmes, racisées, immigrantes, gouines, trans, LGBTQI+, en situation d'handicap, précaire, mères isolées, ...

Leur initiative vise à faire entendre les voix des Femmes victimes quotidiennement de violences sexistes, sexuelles, lesbophobes, hétérosexistes, transphobes, islamophobes, racistes et de classes.

Leur démarche vise à interpeller l'ensemble des partis se portant candidat aux municipales à Amiens.

## B - Technologies utilisées

Pour la création de ce site, j'ai utilisé les technologies suivantes :

### **Wampserver**

Wampserver est une plateforme de développement web permettant de faire fonctionner localement des scripts PHP. C'est un environnement comprenant trois serveurs, un interpréteur de script, ainsi que PHPMyAdmin pour l'administration Web des bases de données.

### **PHPMyAdmin**

PHPMyAdmin est l'une des plus célèbres interfaces pour gérer les bases de données MySQL sur un serveur PHP. Cette interface permet d'exécuter des requêtes comme la création de bases de données, de tables, des requêtes d'insertions, de mises à jours, de suppressions et de modifications, ainsi que l'attribution et la révocation de droits et l'import/export.

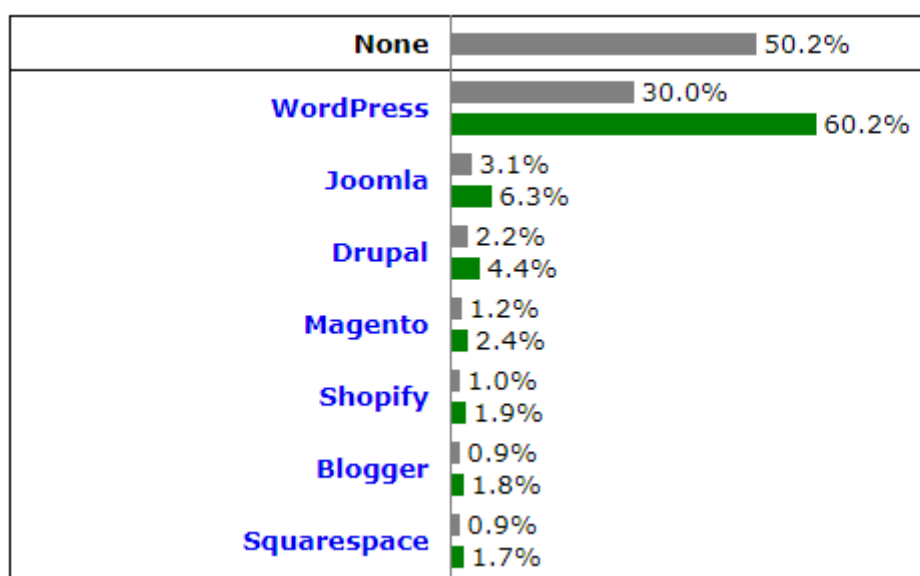
### **WordPress**

WordPress est ce que l'on appelle un système de gestion de contenu (CMS), c'est une plateforme avec laquelle on va pouvoir gérer notre site internet et son contenu en toute autonomie grâce à son back-office.

WordPress permet :

- De créer les articles et pages d'un site internet en y insérant du texte, des images ...
- De gérer les commentaires déposés par les visiteurs.
- D'organiser le contenu grâce à des taxonomies.
- D'intégrer des menus permettant d'accéder directement à certaines pages ou articles.
- D'insérer des widgets, d'ajouter des extensions.
- De choisir le thème (le visuel du site).
- De gérer les utilisateurs en leur affectant des droits.

WordPress est le CMS le plus utilisé au monde, le baromètre W3Techs indique que le CMS représente désormais 30% des sites web mondiaux.



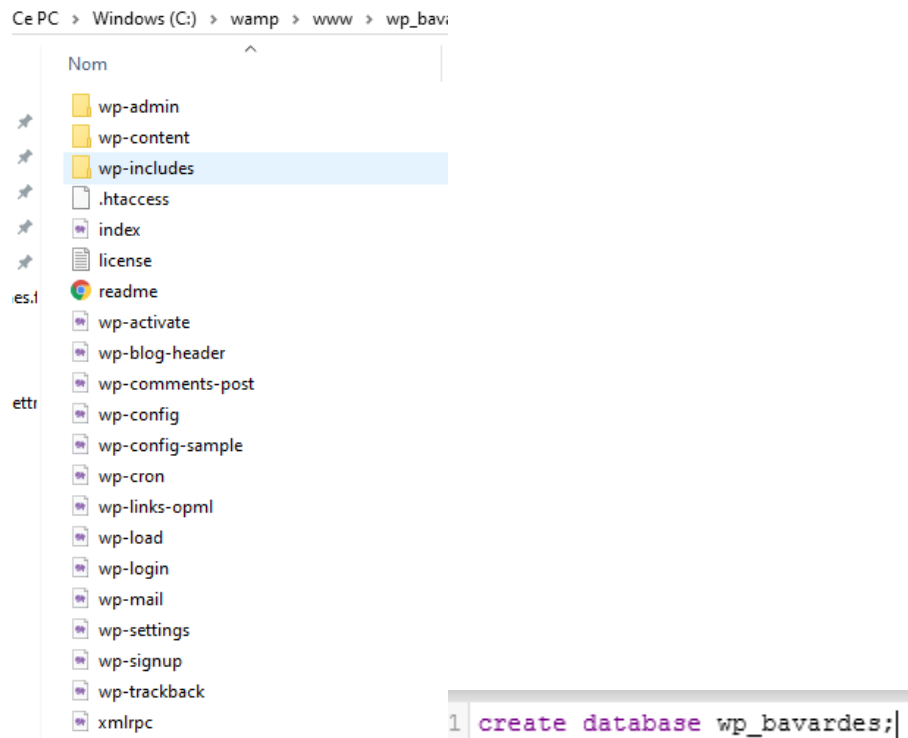
Comment le lire ? Le baromètre donne deux mesures :

- En gris est donné le pourcentage total des sites Web s'appuyant sur le CMS.
- En vert est donné le pourcentage relatif des sites Web s'appuyant sur le CMS.

Dès le départ, le baromètre indique que 50,2% des sites Web n'utilisent aucun système de gestion de contenu. Ce qui veut dire que le pourcentage relatif est calculé sur la portion restante. Ainsi, WordPress est utilisé par 30% de l'ensemble des sites, soit une part de marché de 60,2% des CMS (pourcentage relatif).

## C - Création du site avec WordPress

La création de ce site s'est faite en local.



Après avoir installé WordPress, j'ai renommé le dossier "wp\_bavardes", j'ai ensuite créé la base de données dans PHPMyAdmin du même nom.



En écrivant « localhost/wp\_bavardes » dans la barre d'adresse URL, je suis tout de suite redirigée vers cette page. En cliquant sur « C'est parti ! » je suis dirigée vers la page suivante.

Vous devez saisir ci-dessous les détails de connexion à votre base de données. Si vous ne les connaissez pas, contactez votre hébergeur.

Nom de la base de données: wp\_bavardes. Le nom de la base de données avec laquelle vous souhaitez utiliser WordPress.

Identifiant: root. Nom d'utilisateur MySQL.

Mot de passe: . Votre mot de passe de base de données.

Adresse de la base de données: localhost. Si localhost ne fonctionne pas, demandez cette information à l'hébergeur de votre site.

Préfixe des tables: wp\_. Si vous souhaitez faire tourner plusieurs installations de WordPress sur une même base de données, modifiez ce réglage.

Envoyer


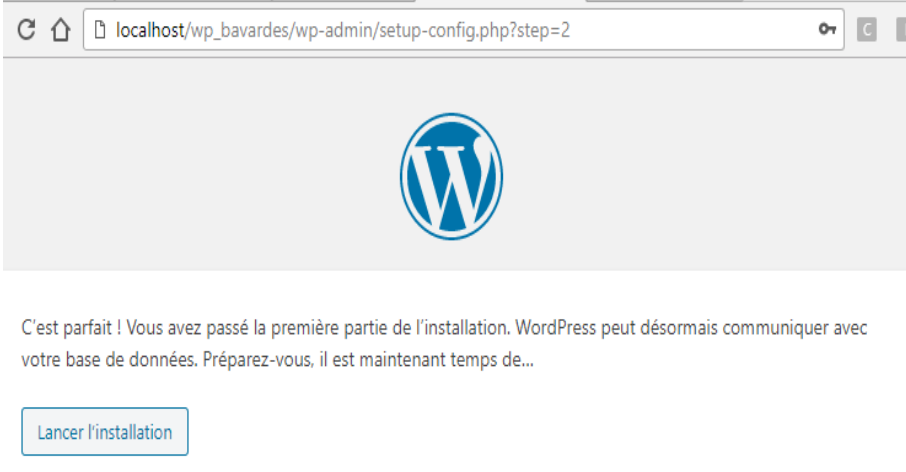
Je renseigne dans cette dernière le nom de la base de données que j'ai créé au préalable, l'identifiant et le mot de passe (les champs de saisie pour l'adresse de la base de données et le préfixe des tables été déjà pré remplis).

En validant le formulaire 12 tables se sont ajoutées dans la base de données

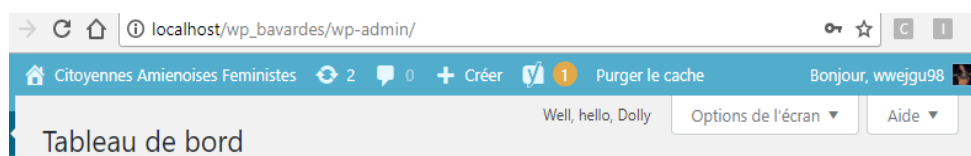
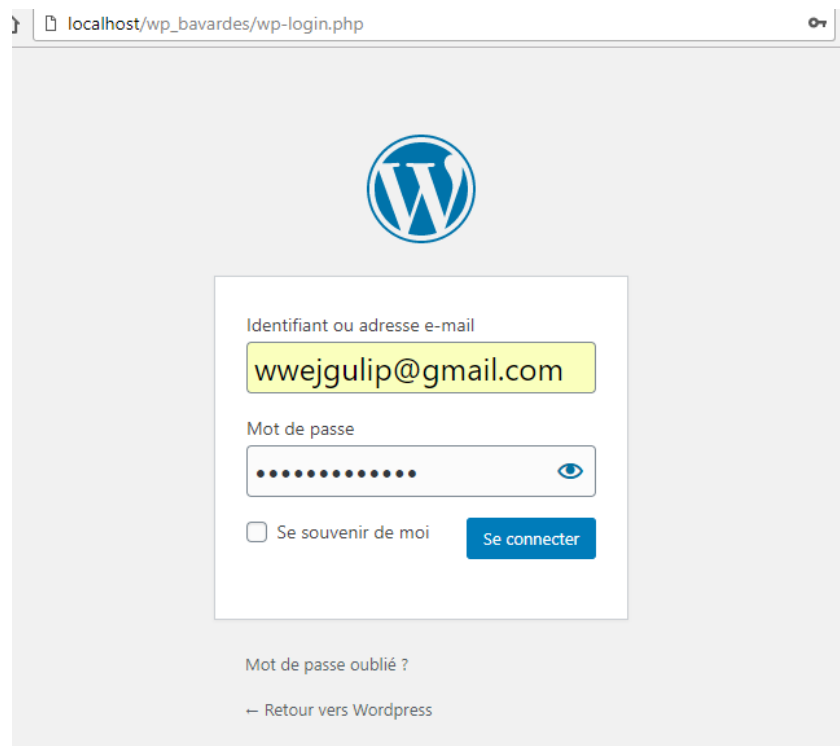
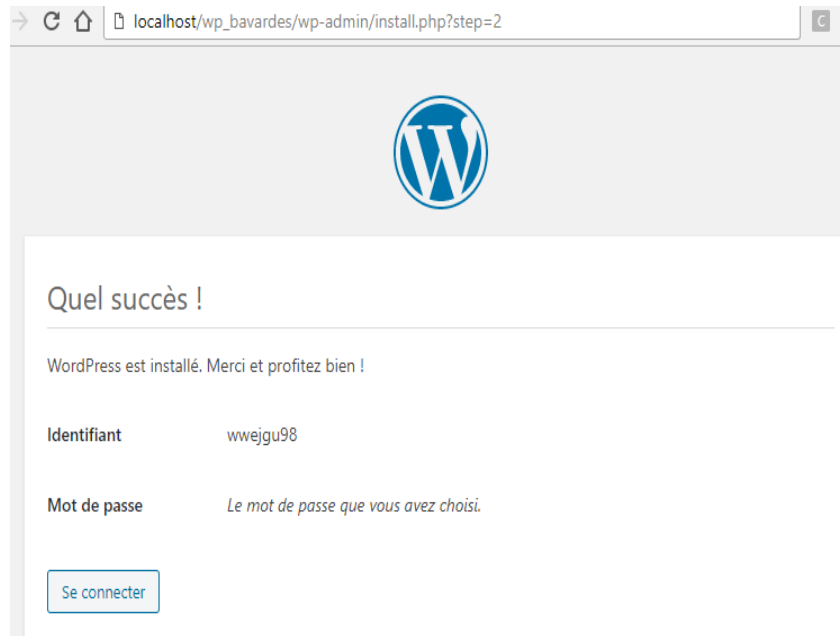
<input type="checkbox"/>	wp_commentmeta	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_comments	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_links	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_options	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_postmeta	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_posts	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_termmeta	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_terms	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_term_relationships	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_term_taxonomy	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_usermeta	★	Parcourir	Structure	Rechercher	Insérer	🖨
<input type="checkbox"/>	wp_users	★	Parcourir	Structure	Rechercher	Insérer	🖨
12 tables		Somme					



Et je suis dirigée vers cette page pour lancer l'installation.



En lançant l'installation s'affiche cette page où je remplis les champs requis (titre du site, identifiant, mot de passe, adresse email) et clique sur "Installer WordPress" qui me renvoie à la page suivante



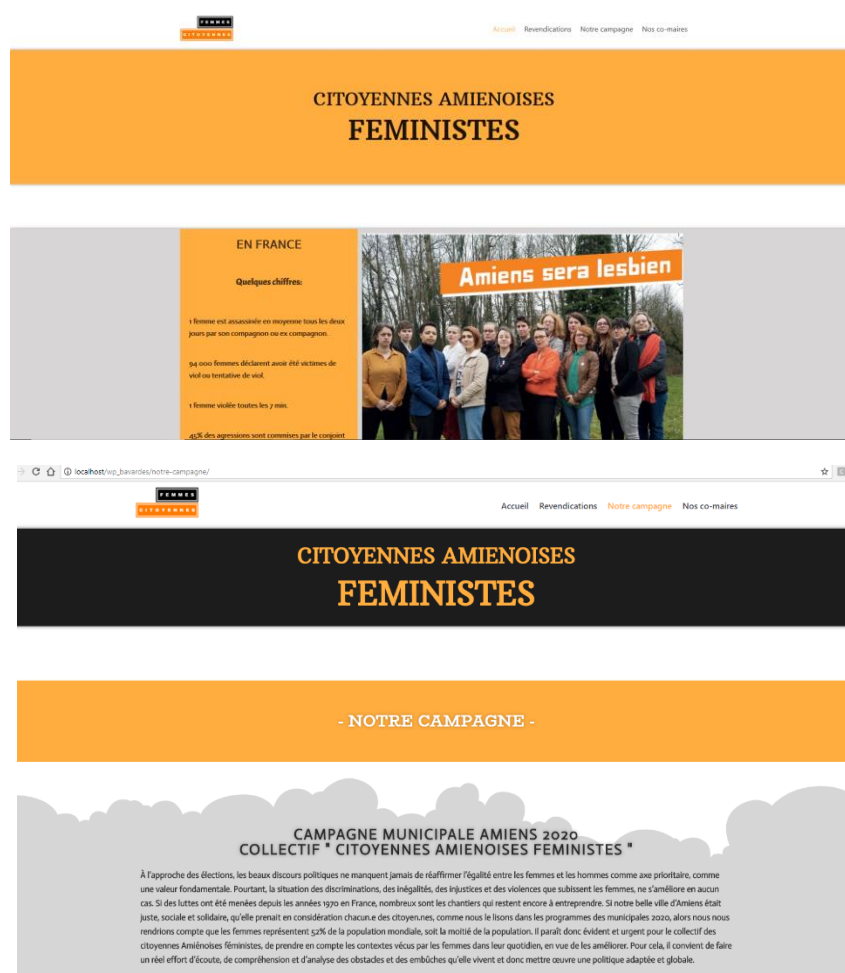
Une fois l'installation finie, je me connecte à WordPress avec mon adresse email (ou mon identifiant) et le mot de passe que j'ai renseigné dans la page précédente pour accéder à mon tableau de bord.

De ce tableau de bord je me suis dirigée dans la rubrique "extensions" depuis laquelle j'ai installé les extensions suivantes :

**iThemes Security** est une extension pour la sécurité.

**Yoast SEO** est la première solution SEO tout-en-un pour WordPress.

**Elementor** est le constructeur de page par glisser-déposer le plus avancé.



## D - Mise en ligne

Pour cette mise en ligne j'ai rencontré beaucoup de difficultés au niveau de la migration de la base de données, lorsque je changeais le nom de domaine local avec celui pour mettre en ligne ça a créé de nombreux problèmes en interne. J'ai donc décidé de refaire totalement le site depuis la plateforme OVH. Le site a été mis en ligne le mercredi 24 juin avec OVH et WinSCP.



## Conclusion

Débutante en développement, mon apprentissage en formation m'a permis de pouvoir apprendre les bases de la programmation. La majorité des projets ont été faits en autonomie complète, malgré les difficultés j'ai toujours trouvé le moyen de résoudre les problèmes que j'ai pu rencontrer.

Ma période de stage m'a permis de mieux comprendre et d'apprécier le monde du développement informatique, notamment avec les projets que le collectif *les Bavardes* m'ont confié.

Je ressors de cette période avec beaucoup de fierté et la satisfaction du travail que j'ai produit et d'avoir pu mettre en ligne deux sites fonctionnels.

Ce stage et cette formation ont été pour moi une expérience très enrichissante à tous les niveaux et me confortent dans ma volonté de travailler dans ce domaine.