

RAPPORT DE STAGE

Du 30 novembre 2020 au 26 février 2021

Développement Web et Web Mobile



SOMMAIRE

| | |
|--|----------------------|
| Remerciements | <i>Page 3</i> |
| Introduction | <i>Pages 4 à 6</i> |
| Présentation de l'association | <i>Page 7</i> |
| Technologies et langages utilisés | <i>Pages 8 à 9</i> |
| Projets | <i>Pages 10 à 22</i> |
| RGPD | <i>Pages 23 à 24</i> |
| Conclusion | <i>Page 25</i> |

Remerciements

Je tiens à remercier toutes les personnes qui m'ont permis de mener à bien cette formation et la période de stage qui la complète.

Tout d'abord, merci aux formateurs de l'Afpa d'Amiens pour leurs enseignements et leurs conseils, ainsi qu'aux stagiaires en développement web pour cette dynamique de groupe qui m'a aidée dans mon apprentissage.

Enfin, merci à l'association Les Bavardes de m'avoir accueillie pour la durée de mon stage. Cette période fut très enrichissante. Merci à Marion Joly et Estelle Camoes pour leur bienveillance et pour m'avoir guidée tout au long de ce stage.

Introduction

L'association Les Bavardes possède déjà un site Internet *lesbavardes.org*. Ce dernier est un site vitrine avec une session administrateur, et se compose de six sections, en plus de l'accueil : *Qui sommes-nous ? L'équipe, Nos événements, Nous écouter, Artistes & intervenant.es, Nos collaborations*.



Le site est multilingue, avec un affichage en français et la possibilité de l'afficher en anglais, grâce aux logos permettant à l'utilisateur de choisir ce qui lui convient.

La gestion se fait avec l'aide de Git et GitHub. Plusieurs branches existent : master, dev, frontSave, jade, et marion. En commençant mon stage, j'y ai ajouté une branche à mon nom sur laquelle j'ai fait tous mes dépôts git. La branche master est utilisée pour les éléments qui sont mis en ligne. J'ai donc régulièrement fait des commit afin de sauvegarder l'évolution de mon travail.

megane
 6 branches
 0 tags
 [Go to file](#)
[Add file](#)
[Code](#)

Switch branches/tags

Find or create a branch...

Branches

Tags

master

dev

frontSave

jade

marion

✓ megane

default

View all branches

8953e31 2 days ago 202 commits

commit CRUD en CI

2 days ago

1er commit CRUD

7 days ago

commit ajout js pour validation

2 months ago

modification espace admin

9 months ago

bdd

3 months ago

ajout page 404

9 months ago

bavardes_site.sql

1er commit CRUD

7 days ago

index.php

site bavarde codeigniter

10 months ago

Help people interested in this repository understand your project by adding a README.

Add a README

```

Admin@DESKTOP-400720G MINGW64 /c:/wamp/www/siteBavardes (megane)
$ git status
On branch megane
Your branch is up to date with 'origin/megane'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   application/controllers/Crud.php
        modified:   application/controllers/Form.php
        modified:   application/models/crud_Model.php
        modified:   application/views/form/contact.php
        modified:   application/views/layouts/footer.php
        modified:   bavardes_site.sql

no changes added to commit (use "git add" and/or "git commit -a")

Admin@DESKTOP-400720G MINGW64 /c:/wamp/www/siteBavardes (megane)
$ git add .

Admin@DESKTOP-400720G MINGW64 /c:/wamp/www/siteBavardes (megane)
$ git commit
hint: Waiting for your editor to close the file...
(electron) Sending uncompressed crash reports is deprecated and will be
[megane 78663d0] modif crud + form
 6 files changed, 10 insertions(+), 8 deletions(-)

Admin@DESKTOP-400720G MINGW64 /c:/wamp/www/siteBavardes (megane)
$ git push
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 1.07 KiB | 1.07 MiB/s, done.
Total 14 (delta 12), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (12/12), completed with 12 local objects.
To https://github.com/Mhyssa/siteBavardes.git
   8953e31..78663d0  megane -> megane

Admin@DESKTOP-400720G MINGW64 /c:/wamp/www/siteBavardes (megane)
$ git status
On branch megane
Your branch is up to date with 'origin/megane'.

nothing to commit, working tree clean

Admin@DESKTOP-400720G MINGW64 /c:/wamp/www/siteBavardes (megane)

```

Afin de répondre aux besoins de l'association, plusieurs missions m'ont été confiées : créer un formulaire de contact pour faciliter la prise de contact entre l'association et de potentiels partenaires, et faire un CRUD afin de gérer le stock de goodies de l'association.

Présentation de l'association

Les Bavardes est une « association lesbienne, bi, trans, queer et hétéro féministe engagée 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. »

Cette association existe depuis juin 2017 et organise diverses actions dans le but de lutter contre le sexisme et les LGBTQphobies.



Technologies et langages utilisés

Figma est un outil de design d'interface collaboratif en temps réel.

Canva est une plateforme de conception graphique qui permet de créer des affiches, des présentations et autres contenus visuels.

Visual Studio Code est un éditeur de code extensible, développé par Microsoft. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétude intelligente du code et Git intégré.

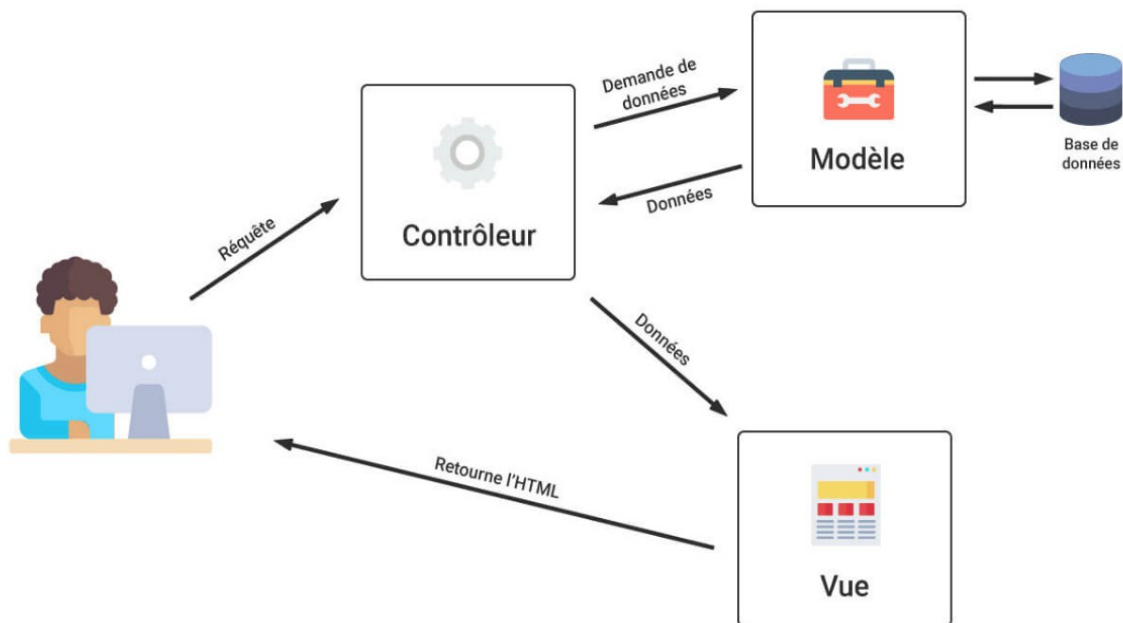
Git est un logiciel de gestion de version permettant de conserver un historique des modifications apportées sur un projet afin de pouvoir rapidement identifier les changements et revenir à une version antérieure en cas de problème.

GitHub est un service en ligne d'hébergement de dépôts Git qui fait office de serveur central pour ces dépôts et permet un travail collaboratif.

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.

PhpMyAdmin est une application Web de gestion pour les systèmes de gestion de base de données MySQL.

CodeIgniter est un framework PHP suivant le motif de conception MVC (Model View Controller). La vue correspond généralement à la page qui sera visible par l'utilisateur. La vue est appelée dans le contrôleur, ainsi que tous les éléments qui seront affichés. Le modèle n'est pas obligatoire, mais permet de gérer les opérations liées à la base de données.



Bootstrap est un framework CSS comportant un ensemble de fichiers CSS et JavaScript fonctionnant ensemble et permettant de créer des designs complexes de manière relativement simple.

PHP (Hypertext Preprocessor) est un langage de programmation libre.

JavaScript est un langage de script léger, orienté objet, principalement connu comme le langage de script des pages Web.

SQL est un langage informatique normalisé servant à exploiter des bases de données. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données.

Projets

I. Formulaire de contact

La première tâche qui m'a été confiée est la création d'un formulaire de contact. Avant de développer le formulaire, j'ai commencé par faire sa maquette sur Figma afin d'avoir une idée plus précise du résultat visé.

1. Maquette du formulaire

La difficulté était de prendre en main un outil qui m'était inconnu. Cependant, les diverses possibilités proposées par Figma permettent d'avoir un rendu détaillé.

Nous contacter

Nom*

Prénom

Mail*

Ville

Objet*

Message*

Envoyer

2. Développement du formulaire

J'ai ensuite développé le formulaire de contact en utilisant CodeIgniter.

Pour cela, j'ai créé le contrôleur Form, et la vue contact.php. Le contrôleur permet d'afficher la vue contact contenant le code du formulaire de contact,

mais aussi la vue header et la vue footer. Ces deux dernières vues existaient déjà.

Le site étant disponible en français et en anglais, j'ai dû adapter mon formulaire en le traduisant afin que les deux versions existent. Le texte, c'est à dire ici les labels pour notre formulaire, est écrit dans les fichiers content_lang.php. Chaque langue a son propre dossier.

```
//formulaire index
$data['form_name'] = $this->lang->line('form_name');
$data['form_firstname'] = $this->lang->line('form_firstname');
$data['form_mail'] = $this->lang->line('form_mail');
$data['form_city'] = $this->lang->line('form_city');
$data['form_object'] = $this->lang->line('form_object');
$data['form_message'] = $this->lang->line('form_message');
$data['form_submit'] = $this->lang->line('form_submit');

//page success
$data['form_success'] = $this->lang->line('form_success');
```

```
<span class="input-group-text" id="inputGroup-sizing-sm"><?= $form_name ?></span>
```

Ainsi, dans la vue au lieu d'écrire 'nom', nous écrivons `<?=$form_name?>` qui permet de l'avoir en français ou en anglais selon la version choisie par l'utilisateur. Dans le contrôleur, nous appelons donc la ligne de cette façon :
`$data['form_name'] = $this->lang->line('form_name');`
L'opération est répétée pour chaque zone de texte.

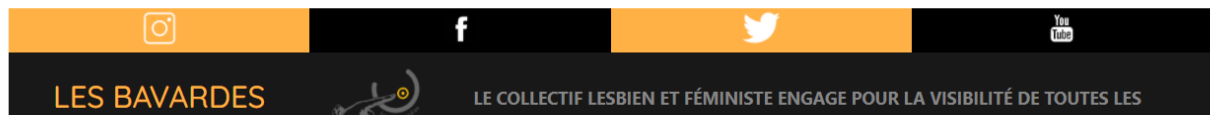
Une fois le formulaire envoyé, si toutes les informations sont valides, l'utilisateur arrive sur une page lui indiquant le succès de l'envoi.

```
if ($this->form_validation->run() == FALSE){
    $this->load->view('form/contact', $data);
} else {
    $this->load->view('form/success', $data);
}
```

Pour cela, j'ai créé une vue success.php indiquant l'envoi du formulaire.



Votre message a bien été envoyé. Merci de votre intérêt !



La mise en page de la vue contact et de la vue success a été faite en Bootstrap. La bannière contact a été faite avec Canva en reprenant le modèle des bannières présentes sur les autres pages afin de garder une cohérence avec le style du site.

3. Validation du formulaire

a. CodeIgniter

La validation de formulaire en CodeIgniter se fait avec le chargement de la librairie form_validation et les set_rules dans le contrôleur.

```
// Chargement des assistants 'form' et 'url'
$this->load->helper(array('form', 'url'));

//Chargement de la librairie 'validation'
$this->load->library('form_validation');
```

```
//if($this->input->post('contactForm')){
    $this->form_validation->set_rules( 'nom', 'nom' , 'required');
    $this->form_validation->set_rules( 'mail', 'mail' , 'required');
    $this->form_validation->set_rules( 'objet', 'objet' , 'required');
    $this->form_validation->set_rules( 'message', 'message' , 'required');

    if ($this->form_validation->run() == FALSE){

        $this->load->view('form/contact', $data);
    } else {

        $this->load->view('form/success', $data);
    }
}
```

La méthode `set_rules` prend en compte trois paramètres :

- Le champ name,
- La valeur qui va être utilisée pour l'affichage de l'erreur selon le champ concerné,
- Les règles de validation qu'on applique sur notre champ.

Ici, nous avons utilisé `required` comme troisième paramètre afin de rendre obligatoire la saisie dans le champ.

Nous avons ensuite créé une condition. Si la validation du formulaire renvoie faux, la vue `contact` est affichée, sinon la vue `success` est affichée. En d'autres termes, si la validation du formulaire renvoie faux l'utilisateur sera de nouveau sur le formulaire de contact afin de le compléter correctement, alors que dans le cas où les données sont validées, l'utilisateur sera dirigé vers la page `success` qui lui indiquera que son message a bien été envoyé.

b. JavaScript

Afin de valider le formulaire en JavaScript, j'ai commencé par créer un fichier `contact.js` dans lequel j'ai d'abord déclaré les variables correspondant à mon formulaire.

```
let formValid = document.getElementById('submit');

let cnom = document.getElementById('nom');
let cprenom = document.getElementById('prenom');
let cmail = document.getElementById('mail');
let cville = document.getElementById('ville');
let cobjet = document.getElementById('objet');
let cmessage = document.getElementById('message');
```

La première variable correspond à l'envoi du formulaire, les autres variables correspondent aux champs du formulaire de contact.

Nous utilisons `getElementById` afin de récupérer les champs du formulaire de contact.

```
// définition du ciblage pour l'affichage du message d'erreur
let missCnom = document.getElementById('missNom');
let missCmail = document.getElementById('missMail');
let missCobjet = document.getElementById('missObjet');
let missCmessage = document.getElementById('missMessage');
```

J'ai ensuite défini des variables pour les messages d'erreur lors de la non saisie d'un champ, en utilisant toujours le `getElementById` pour la récupération des champs. Pour cela, j'ai ajouté un `span` id avec le nom de la variable correspondant au champ, dans la vue contact.

J'ai ensuite défini des expressions régulières afin de vérifier que l'utilisateur remplit le formulaire correctement.

```
let cnomValid = /^[A-ZÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙÚÛÜÝŸÆ]
[a-zàáâãäåçèéêëìíîïñóôõöùúûüýÿæ]*([a-zAÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙÚÛÜÝŸÆ]
([A-ZÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙÚÛÜÝŸÆ][a-zàáâãäåçèéêëìíîïñóôõöùúûüýÿæ]*[a-zAÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÑÒÓÔÕÖÙÚÛÜÝŸÆ]+))*$/
```

Par exemple, pour le nom, il doit commencer par une lettre majuscule, pouvant avoir un accent, ça peut être un nom composé avec un tiret, sans être obligatoire, et en dehors du tiret il ne doit comporter que des lettres accentuées ou non.

Une fois toutes ces variables définies, nous pouvons passer à la validation du formulaire de contact.

```
formValid.addEventListener('click', validation);

function validation(event) {
  //vérification du champ nom

  //si le champ est vide
  if (cnom.validity.valueMissing) {
    event.preventDefault();
    missNom.textContent = 'Renseigner ce champ /Complete this field';
    missNom.style.color = 'red';
  }

  //si le format de données est incorrect
  else if (cnomValid.test(cnom.value) == false) {
    event.preventDefault();
    missCnom.textContent = 'Format incorrect / wrong formt';
    missCnom.style.color = 'orange';
  }
  else {
    missCnom.textContent = 'Ok !';
    missCnom.style.color = 'green';
  }
}
```

La validation du formulaire se fait à partir du moment où l'utilisateur clique sur

le bouton 'Envoyer'.

La fonction validation va indiquer un message d'erreur en rouge si le champ n'est pas rempli, un message d'erreur en orange si le format est incorrect par rapport aux expressions régulières définies précédemment, et un message en vert si le champ est correctement rempli.

Si tous les champs sont correctement remplis, l'utilisateur est dirigé vers la page success.

II. Mise à jour du site

Cette deuxième mission consistait à mettre à jour la page d'accueil en changeant les images à la une afin de mettre celles correspondant aux nouveaux événements.

Pour cela, j'ai récupéré les images sur les événements Facebook créés par la page de l'association. Il y avait trois nouveaux événements à afficher : Femmes de sport, Femmes de sport blabla et la réunion des bénévoles afin de préparer le 8 mars.

Les images sont rangées dans assets/img/index/alaune.

Les images forment une bannière de trois colonnes, chaque colonne étant un article. Je devais modifier la première et la troisième colonne.

```
----- ARTICLE 3 -----
----->

<div class="col-12 col-sm-12 col-md-4 bgpastel">
  <a href="https://www.facebook.com/events/428648841663955/" target="_blank">
    
  </a>
  <a href="https://www.facebook.com/events/168774534586049" target="_blank">
    
  </a>
</div>
</div>
```

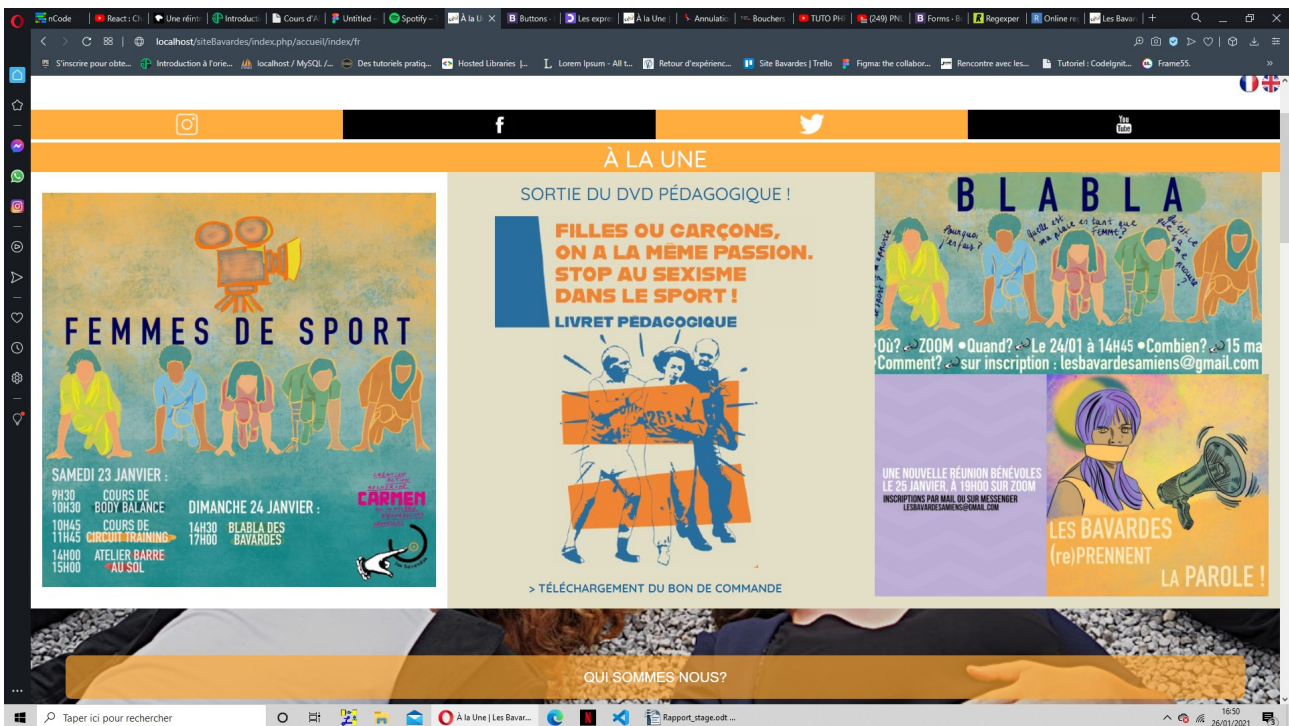
Afin de mettre l'accueil à jour, les modifications se faisaient sur la source, le lien et la balise alt de chaque image.

Le lien permet d'aller directement sur la page Facebook de l'événement correspondant à l'image, et ainsi avoir toutes les informations nécessaires

pour y participer.

Pour optimiser le référencement de l'image, j'ai modifié la balise alt de façon à décrire l'événement concernant chaque image. Cette balise, aussi appelée texte alternatif, est le texte lu par les moteurs de recherche. C'est aussi ce qui remplacera l'image si le navigateur n'arrive pas à l'afficher.

L'utilisation de Bootstrap permet un affichage responsive des images. Un site Web est dit responsive lorsque sa consultation est confortable sur des écrans de tailles très différentes.



Page d'accueil du site Les Bavardes

III.CRUD

CRUD signifie create, read, update et delete (écrire, lire, mettre à jour, et supprimer). Il s'agit des opérations de base attendues par un système de gestion de base de données (mais pas uniquement).

Cette troisième mission consiste en la création d'une base de données. Celle-ci contiendra les affiches et badges, afin de les mettre en vente plus tard, ainsi que des livres, des jeux, des BD, et des magazines. La base de données permettra de gérer le stock de ces derniers afin de les prêter et ainsi constituer une bibliothèque féministe.

1. Création de la table affiches

La première étape a donc été la création de la table Affiches. Pour cela, j'ai utilisé phpMyAdmin et SQL.

```
/*Création de la table Affiches*/

DROP TABLE IF EXISTS affiches;
CREATE TABLE affiches (
  aff_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  aff_nom VARCHAR(50) NOT NULL,
  aff_stock INT NOT NULL,
  aff_description VARCHAR(100) NOT NULL,
  aff_prix INT
);
```

La table Affiches comporte l'ID pour identifier les affiches, le nom, le stock, la description et le prix. L'auto_increment permet que la colonne ID s'incrémente automatiquement à chaque nouvel enregistrement d'une affiche.

| # | Nom | Type | Interclassement | Attributs | Null | Valeur par défaut | Commentaires | Extra | Action |
|--------------------------|-----|-----------------|-----------------|-----------|------|-------------------|--------------|----------------|---------------------------|
| <input type="checkbox"/> | 1 | aff_id | | | Non | Aucun(e) | | AUTO_INCREMENT | Modifier Supprimer Plus |
| <input type="checkbox"/> | 2 | aff_nom | utf8_general_ci | | Non | Aucun(e) | | | Modifier Supprimer Plus |
| <input type="checkbox"/> | 3 | aff_stock | | | Non | Aucun(e) | | | Modifier Supprimer Plus |
| <input type="checkbox"/> | 4 | aff_description | utf8_general_ci | | Non | Aucun(e) | | | Modifier Supprimer Plus |
| <input type="checkbox"/> | 5 | aff_prix | | | Oui | NULL | | | Modifier Supprimer Plus |

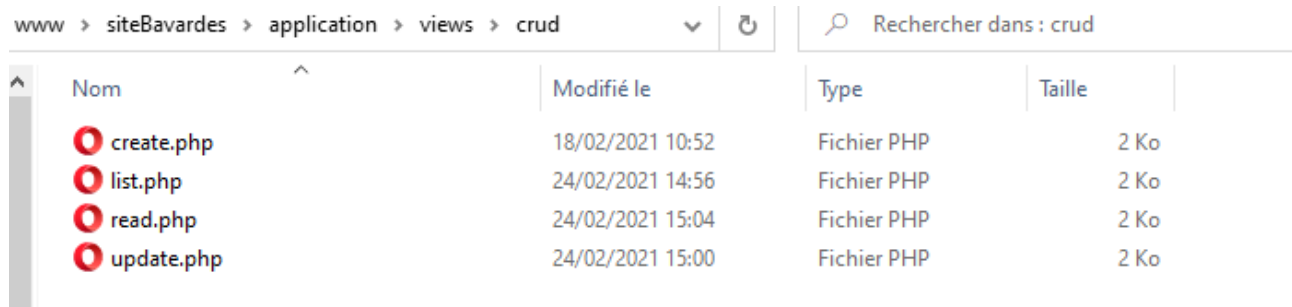
Tout cocher Avec la sélection : Parcourir Modifier Supprimer Primaire Unique Index Texte entier Texte entier

2. Création du CRUD

Pour faire un CRUD avec CodeIgniter, j'ai commencé par créer les routes.

```
$route['crud'] = "crud/index";
$route['crud/(:num)'] = "crud/read/$1";
$route['crudCreate']['post'] = "crud/store";
$route['crudEdit/(:any)'] = "crud/edit/$1";
$route['crudUpdate/(:any)']['put'] = "crud/update/$1";
$route['crudDelete/(:any)']['delete'] = "crud/delete/$1";
```

J'ai ensuite créé le contrôleur Crud.php dans lequel les différentes vues sont appelées : list, create, read, update.



| www > siteBavardes > application > views > crud | | | | |
|---|------------------|-------------|--------|--|
| Rechercher dans : crud | | | | |
| Nom | Modifié le | Type | Taille | |
| create.php | 18/02/2021 10:52 | Fichier PHP | 2 Ko | |
| list.php | 24/02/2021 14:56 | Fichier PHP | 2 Ko | |
| read.php | 24/02/2021 15:04 | Fichier PHP | 2 Ko | |
| update.php | 24/02/2021 15:00 | Fichier PHP | 2 Ko | |

La vue list présente sous forme d'un tableau les affiches présentes dans la table de données. C'est aussi sur cette vue qu'il est possible de supprimer un élément.

La vue create contient un formulaire qui permet d'ajouter de nouveaux éléments à la base de données. Dans notre cas, elle permet d'ajouter de nouvelles affiches.

La vue read permet de voir les informations d'une affiche, grâce aux fonctions du modèle qui récupère les informations dans la base de données.

La vue update contient également un formulaire, mais celui-ci permet de faire des modifications dans les informations déjà enregistrées dans la base de données.

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

    $this->load->library('form_validation');
    $this->load->library('session');
    $this->load->model('crudModel');

    $this->crud = new CrudModel;
}
```

Dans le contrôleur, nous chargeons d'abord les librairies pour la validation de formulaire et la session, puis le modèle créé pour le CRUD. La méthode `construct` permet d'appliquer ces éléments à l'ensemble du contrôleur.

```
public function read($id)
{
    $item = $this->crud->find_item($id);
    $this->load->view('crud/read',array('item'=>$item));
}
```

Ensuite, nous utilisons les fonctions du modèle afin d'effectuer les actions demandées, et nous chargeons la vue correspondante. Dans le cas de la vue `read`, nous cherchons à afficher les informations d'une affiche par exemple.

```

<div class="row">
  <div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
      <strong>Nom</strong>
      <?php echo $item->aff_nom; ?>
    </div>
  </div>

  <div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
      <strong>Description</strong>
      <?php echo $item->aff_description; ?>
    </div>
  </div>

  <div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
      <strong>Stock</strong>
      <?php echo $item->aff_stock; ?>
    </div>
  </div>

  <div class="col-xs-12 col-sm-12 col-md-12">
    <div class="form-group">
      <strong>Prix</strong>
      <?php echo $item->aff_prix; ?>
    </div>
  </div>
</div>

```

La vue permet d'afficher les données récupérées par le modèle et le contrôleur.

Les données sont vérifiées grâce à l'utilisation des `set_rules` dans le contrôleur.

RGPD

I. Définition et mise en place

Dans le cadre associatif, j'ai eu l'occasion d'assister à la visio-conférence sur le Retour d'expérience de la mise en conformité du RGPD dans une association. La conférence avait lieu sur Zoom et était animée par Madame Hamra Rana, co-fondatrice de Humanity Diaspo (ONG humanitaire).

Le RGPD est le Règlement Général de la Protection des Données. Son but est d'encadrer le traitement des données personnelles sur le territoire de l'Union Européenne.

L'Union Européenne a souhaité réglementer le traitement des données personnelles, notamment suite aux abus des GAFAM (Google, Apple, Facebook, Amazon, Microsoft).

La CNIL veille à faire respecter le RGPD. La CNIL considère plusieurs précautions à prendre comme étant élémentaires :

1. Faire le point sur comment sont traitées les données.

2. Définir qui a accès à ces données dans votre structure.

→ Les anciens membres du bureau ou salariés ne doivent plus y avoir accès.

3. Faire un tri.

→ Un bénévole ou adhérent qui n'est plus actif depuis des années ne doit plus figurer dans vos données.

4. Consentement des propriétaires de ces données.

→ Obligation d'informer les propriétaires sur le traitement des données personnelles.

Important : si la base de données a été piratée, la CNIL et les personnes concernées doivent en être informées dans les 72h.

5. Sécuriser.

→ Prévoir un verrouillage automatique de session.

→ Installer un 'pare-feu'.

→ Installer un antivirus et le mettre régulièrement à jour.

→ Mettre en place des mises à jour de sécurité automatiques.

→ Favoriser le stockage de données des utilisateurs sur un espace de stockage.

Les sanctions peuvent aller d'une amende jusqu'à une peine d'emprisonnement, et les deux peuvent être cumulatives.

II. Concernant le site les Bavardes

Toutes les informations relatives au RGPD se trouvent dans la page Politique de confidentialité, celle-ci étant accessible à tous les utilisateurs du site.

Cette page indique les responsables de publication, la gestion des données personnelles avec notamment la durée de conservation et les destinataires des données. Les informations concernant l'utilisation des cookies, et les données recueillies via les formulaires sont aussi présentes sur cette page.

La collecte des données personnelles est limitée au strict nécessaire dans le cadre définit dans la politique de confidentialité, c'est à dire lors d'une prise de contact avec l'association ou encore par la consultation et les actions sur le site dans le cas des cookies. Les données personnelles sont conservées le temps de traitement de la demande, et les cookies sont conservés pour une durée de 13 mois maximum. Ceux-ci permettent d'améliorer la qualité et l'ergonomie du site Internet.

Les personnes pouvant accéder à ces données sont bien définies grâce à un système d'administrateur.

Conclusion

Pour conclure, je peux dire que cette période de stage a été enrichissante, tout autant que la formation qui l'a précédée. Le métier de développeur web nécessite un apprentissage constant, et ces 8 derniers mois m'ont permis d'en découvrir une partie.

Les difficultés rencontrées durant ce stage ont été principalement de prendre en main un site créé par d'autres développeurs, c'était quelque chose d'inédit pour moi du fait qu'en formation je travaillais exclusivement sur mon propre code. De plus, je ne connaissais que les bases de CodeIgniter, j'ai donc dû approfondir mes connaissances afin de mener à bien mes projets.

Je sors de cette période de stage avec encore plus de motivation pour continuer à apprendre et travailler dans ce domaine.