Concepteur Développeur d'Applications

# RAPPORT DE STAGE

Du 24 mai au 20 août 2021

Stagiaire : Innocent Binyoma

Tuteur : Marion Joly

Formateurs : MM. Sipiere, Cousin et Bruce

# Sommaire

# Remerciements

Je remercie les membres de l’association les Bavardes et particulièrement Marion joly qui m’a accompagné tout au long du stage.

Mes remerciements vont également aux formateurs pour leurs précieux conseils.

# INTRODUCTION

Du 24 mai au 20 août j'ai effectué un stage au sein de l'association les Bavardes dans le cadre de ma formation « Concepteur développeur d'Applications ». Les Bavardes est une association qui cherche à visibiliser les femmes et à lutter contre les préjugés, les violences et les discriminations faites à toutes les femmes[1].

Les deux premières semaines ont été consacrées à la découverte de l'association et de leur site internet. Le stage s'effectuant en télétravail, la découverte s'est faite à travers la lecture des deux rapports de stage des précédents stagiaires développeuses web qui ont contribué au développement du site internet et un mémoire de master 2 « *S'emparer de la cause du droit des femmes et des lesbiennes depuis l'expérience d'un collectif militant dans une ville moyenne des Hauts-de-France »*. Ce dernier m'a permis de saisir l'association dans le contexte socio-politique et militant.

La première mission était la mise à jour et en ligne du site internet de la troisième marche des fiertés à Amiens[2]du 19 juin 2021. Ensuite c'était la conception et le développement de la gestion des affiches, goodies et d'emprunts de la « Bibliothèque Bavardes ». Et enfin le développement d'une application mobile sur flutter.

Dans ce rapport, je reviendrai sur la présentation de l'association, de leur projet du site internet et les réalisations effectuées lors de mon stage.

[1] https://lesbavardes.org/index.php/about/index/fr
[2] http://www.fiertesamiens.fr/

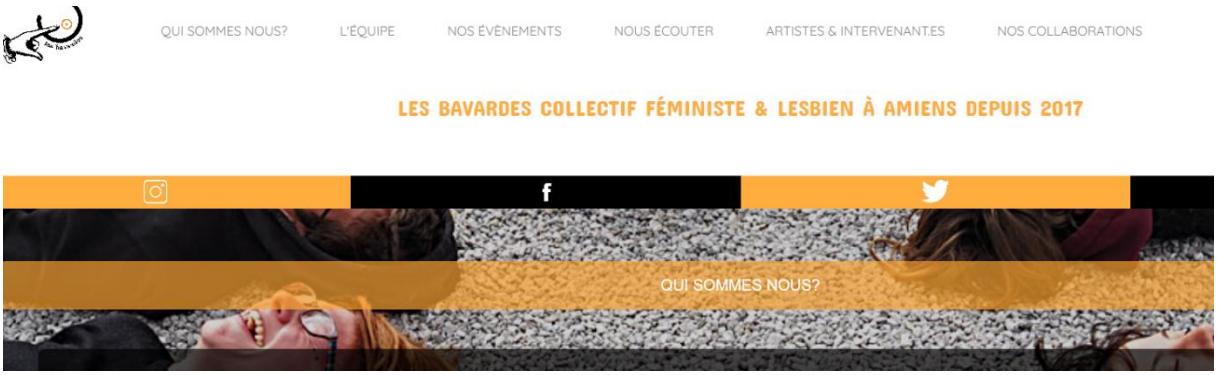
# 1. PRESENTATION DE L'ASSOCIATION ET DU PROJET

## 1. 1. Les Bavardes

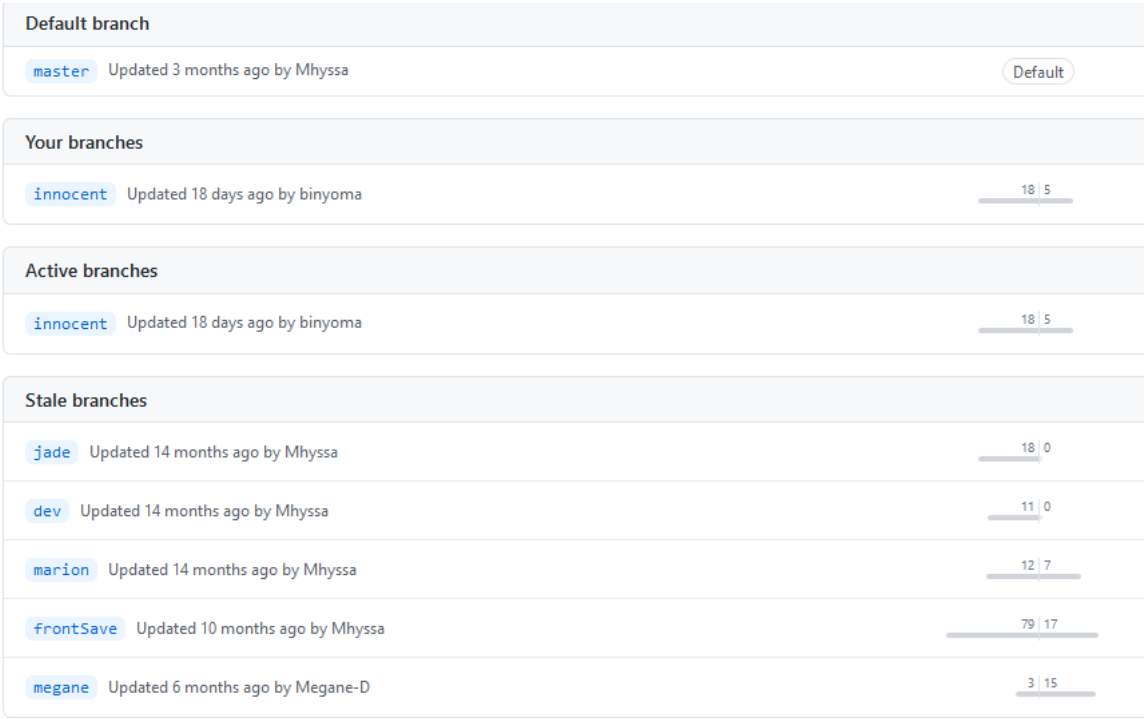
Les Bavardes est un collectif lesbien et féministe qui existe sur Amiens depuis juin 2017. C'est aussi un collectif bi, trans, queer et hétéro, engagé pour la visibilité de toutes les femmes au travers d'espaces et d'actions culturelles et artistiques, ouverts à toutes et à tous, visant à agir, se rencontrer, 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.

## 1.2. Site internet les bavardes

Le site fournit des informations sur le collectif et l'équipe, des liens aux événements et aux podcasts, les artistes et intervenants et leurs collaborations.




Il est développé en PHP avec le framework MVC CodeIgniter. La gestion du projet se fait à l'aide de Git et GitHub. Avant la création de ma branche il y avait 6 branches : master, dev, jade, marion, megane et frontSave.


Default branch
master Updated 3 months ago by Mhyssa
Default
Your branches
innocent Updated 18 days ago by binyoma
18 5
Active branches
innocent Updated 18 days ago by binyoma
18 5
Stale branches
jade Updated 14 months ago by Mhyssa
18 0
dev Updated 14 months ago by Mhyssa
11 0
marion Updated 14 months ago by Mhyssa
12 7
frontSave Updated 10 months ago by Mhyssa
79 17
megane Updated 6 months ago by Megane-D
3 15

# 2.REALISATIONS

## 2.1. Mise à jour de fiertesamiens.fr

La troisième marche des fiertés s'est déroulé le 19 juin 2021. Ma mission consistait à mettre à jour le site qui avait encore les données de la deuxième marche. Il fallait changer entre autres la date, l'affiche et les organisateurs. Vu le thème et le public visé, j'ai conçu le compte à rebours avec les couleurs du drapeau LGBT.


Le site a connu un pic de fréquentations vers le 13 juin comme le montre les chiffres de Google Analytics ci-dessous. Les chiffres étaient au-dessus des attentes d'une des membres de l'association. Ce qui s'explique en partie par la réussite de la troisième marche des fiertés sur Amiens. Aussi le site occupe la première position sur le moteur de recherche google avec les mots-clés « pride Amiens » ou « fiertés Amiens ».

Pendant cette période j'ai aussi pu bénéficier des ateliers en webinaires sur le référencement des associations organisés par l'association *SEO for change*.

Analytics fiertesamiens Toutes les données du sit...

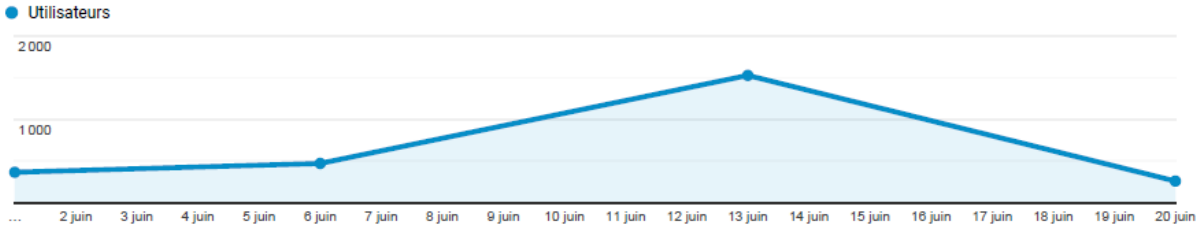
Accéder au rapport

## Présentation de l'audience

Tous les utilisateurs
100,00 %, Utilisateurs

1 juin 2021 - 23 juin 2021

Vue d'ensemble




| | | |
|---|---|---|
| Utilisateurs<br>2 363 | Nouveaux utilisateurs<br>2 287 | Sessions<br>3 732 |
| Nombre de sessions par utilisateur<br>1,58 | Pages vues<br>4 490 | Pages/session<br>1,20 |
| Durée moyenne des sessions<br>00:01:08 | Taux de rebond<br>85,50 % | |


| | Ville | Utilisateurs | % Utilisateurs |
|---|---|---|---|
| 1. | Amiens | 760 | 28,16 % |
| 2. | Lille | 658 | 24,38 % |
| 3. | Paris | 450 | 16,67 % |
| 4. | Abbeville | 66 | 2,45 % |
| 5. | (not set) | 36 | 1,33 % |
| 6. | Saint-Quentin | 31 | 1,15 % |
| 7. | Beauvais | 27 | 1,00 % |

## 2.2. Lesbavardes.org

### 1 . Le cahier des charges

**L'existant**

Sur le site lesbavardes.org il existe une partie back office accessible via le formulaire d'inscription "espace membre". Il y a deux rôles, l'user et l'Admin.

User => Peut créer un espace membre (formulaire de connexion sécurisé, mot de passe hashé ). L'utilisateur·ice·s membre ne reçoit pas de mail de confirmation à l'inscription ( pour le moment ).

Admin => L'administrateur·ice à la possibilité d'ajouter, de supprimer et de modifier un événement, un article, un·e artiste, un podcast ou un partenaire.
L'Admin accède à la liste des utilisateur·ice·s inscrit·e·s
Ne peut pas, pour le moment, directement basculer un utilisateur en admin ( connexion à la base OVH requise )

**La demande**

**Gestion du stock "Affiches et goodies"**

L'association dispose d'affiches disponibles à la vente sur les stands (pas de vente en ligne cela demanderait trop de logistique)

Le User membre du site à la possibilité de consulter les affiches et de faire une demande de réservation à venir chercher en main propre lors d'un prochain événement. L'affiche est à 10 euros. Lors d'une adhésion (5 euros), l'affiche est à 5 euros.

L'Admin pourra supprimer ou ajouter une nouvelle affiche. Il pourra modifier une affiche déjà existante (la photo, le libellé, la quantité, le thème, la réservation possible ou non)

**Gestion d'emprunt "Bibliothèque Bavardes"**

L'association dispose via différents achats et différents dons de livres, de jeux et de magazines.

Le User membre du site à la possibilité de mettre une option sur l'emprunt d'un livre, d'un lot de magazine ou d'un jeu.
Il.elle pourra le récupérer lors d'un prochain événement ou via un membre de l'association.

L'Admin pourra supprimer ou ajouter un article. Il pourra modifier un article déjà existant ( la photo, le libellé, la quantité, la réservation possible ou non )

L'Admin pourra consulter la fiche d'un article qui contiendra différentes informations.
La date d'emprunt, le nom et prénom de l'emprunteur, l'adresse postale, mail et/ou le numéro de téléphone.

## Le travail à effectuer

**UML**

À l'aide du cahier des charges, modéliser les Cas d'utilisation, le diagramme d'activité et le diagramme de classes.

**MCD**

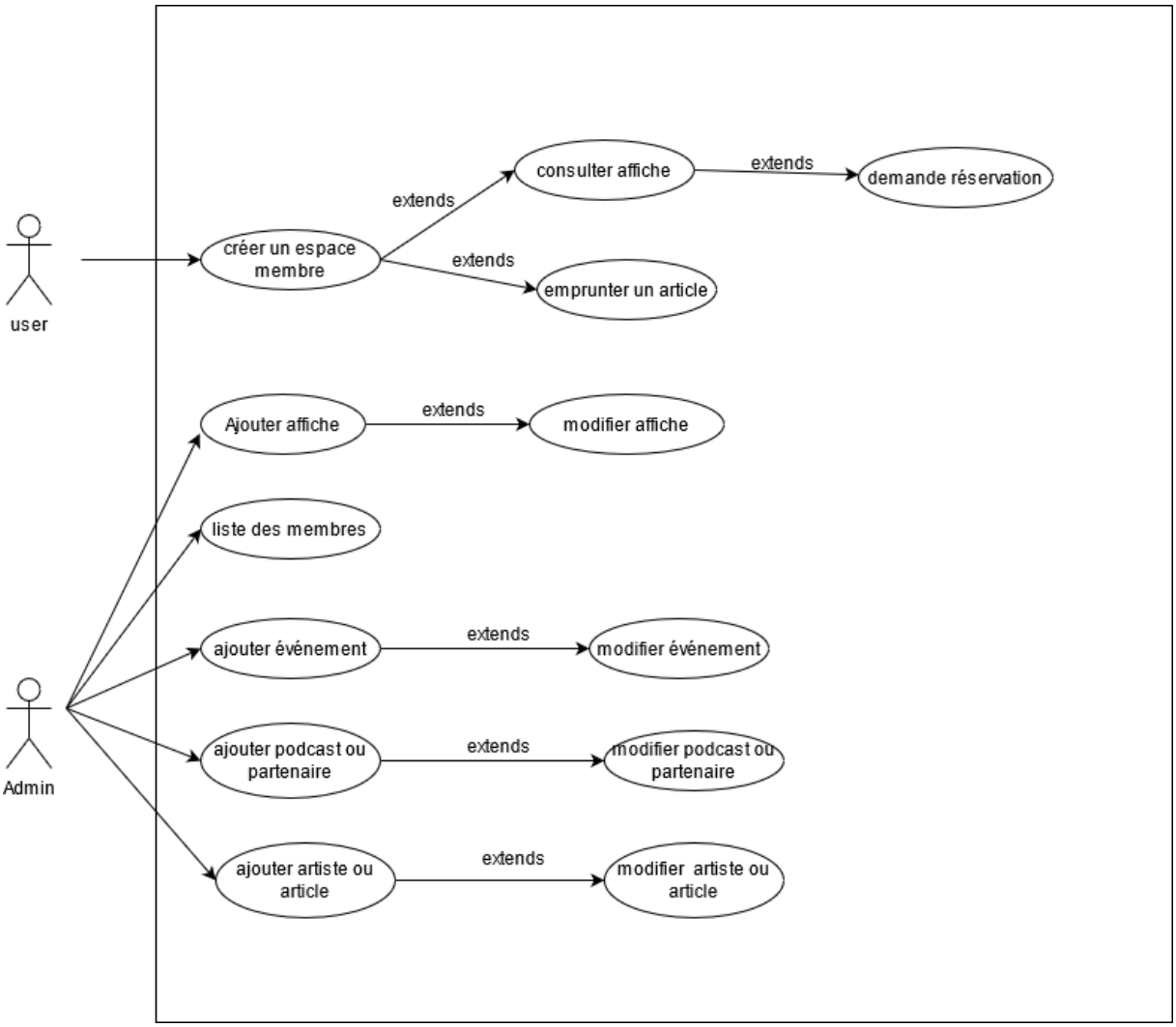
À l'aide du cahier des charges et de l'existant, créer le modèle conceptuel des données grâce à un logiciel de type Merise.

**BDD**

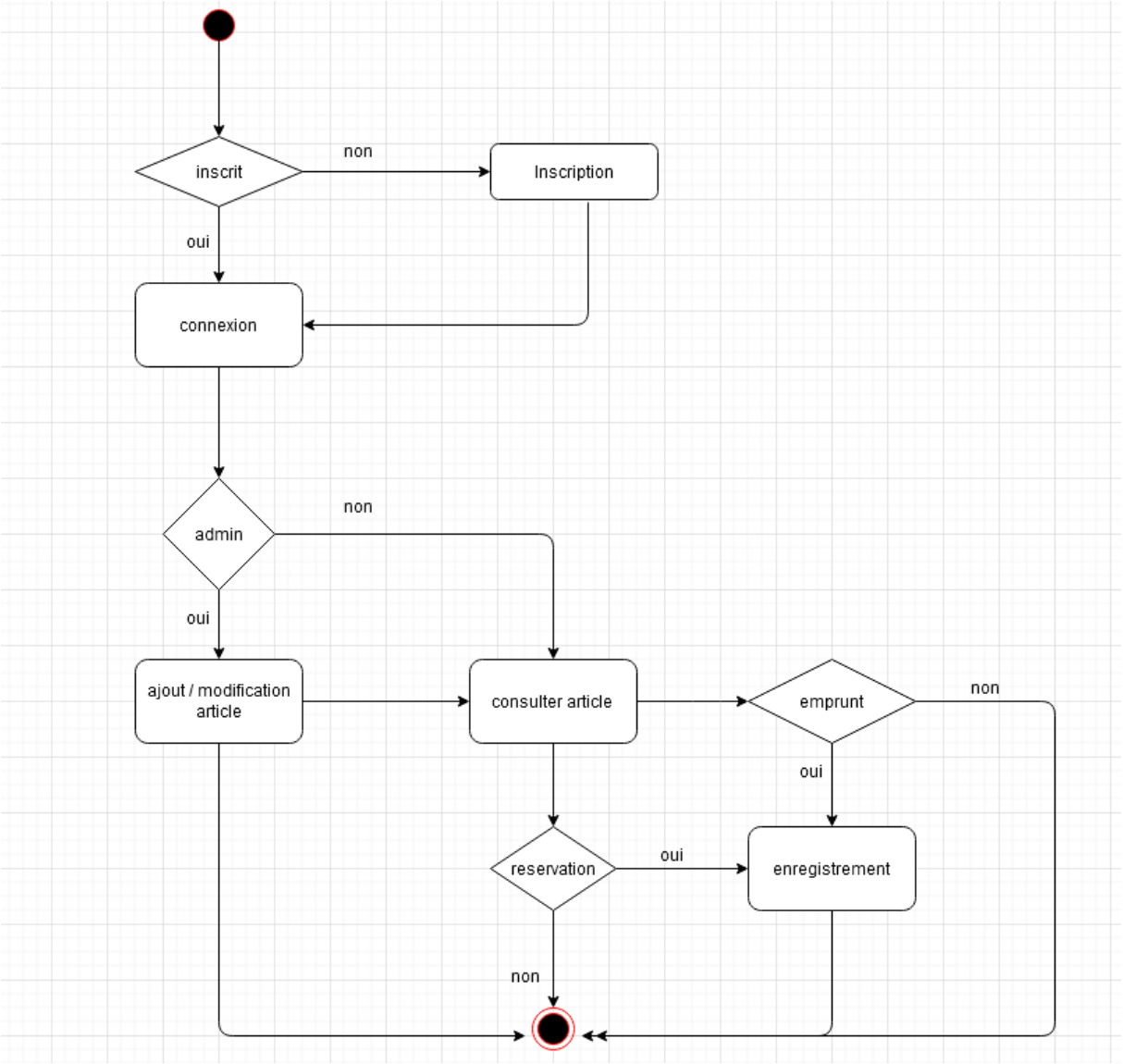
Une fois le mcd validé.
Création en complétant le premier script SQL en alimentant la base des données déjà existantes des affiches et des articles.

1. UML

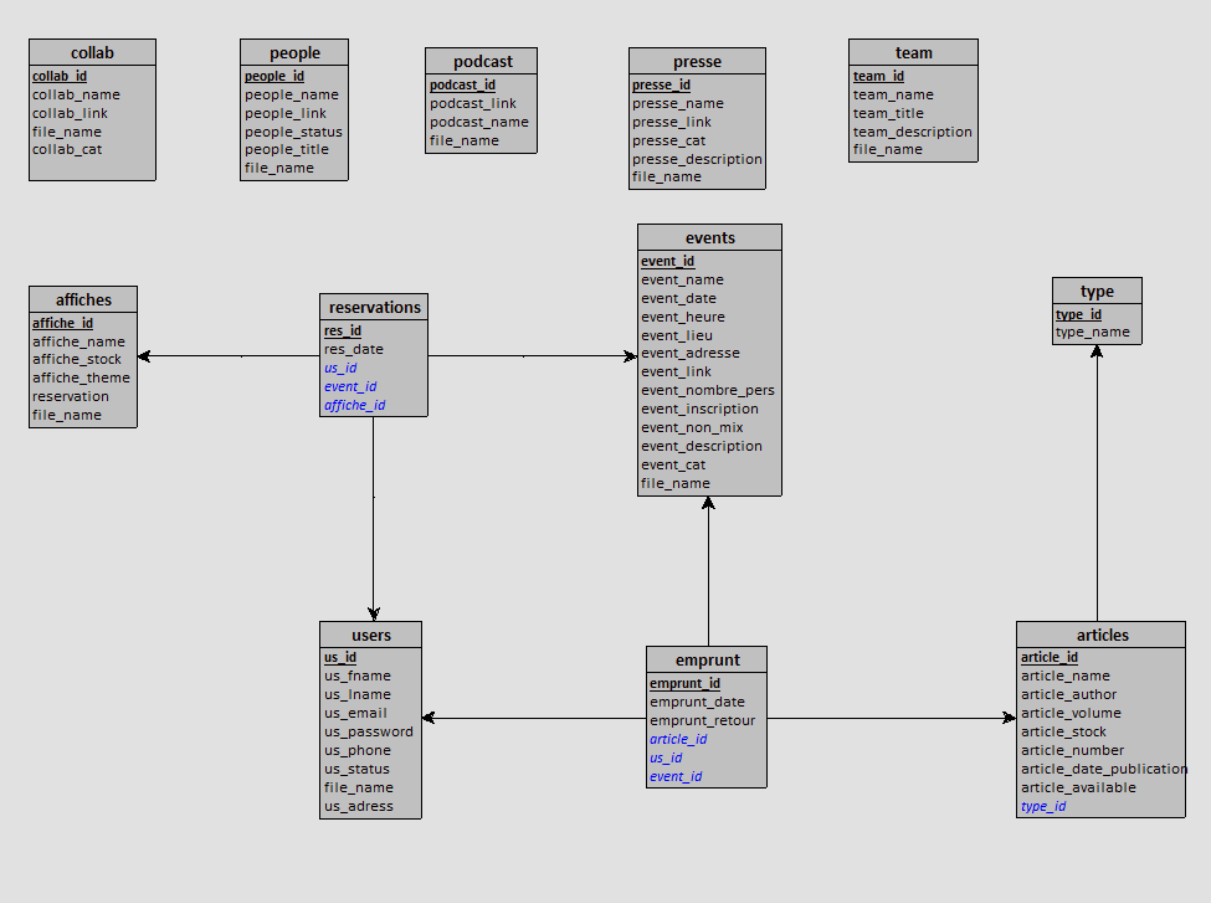
a. Cas d'utilisation



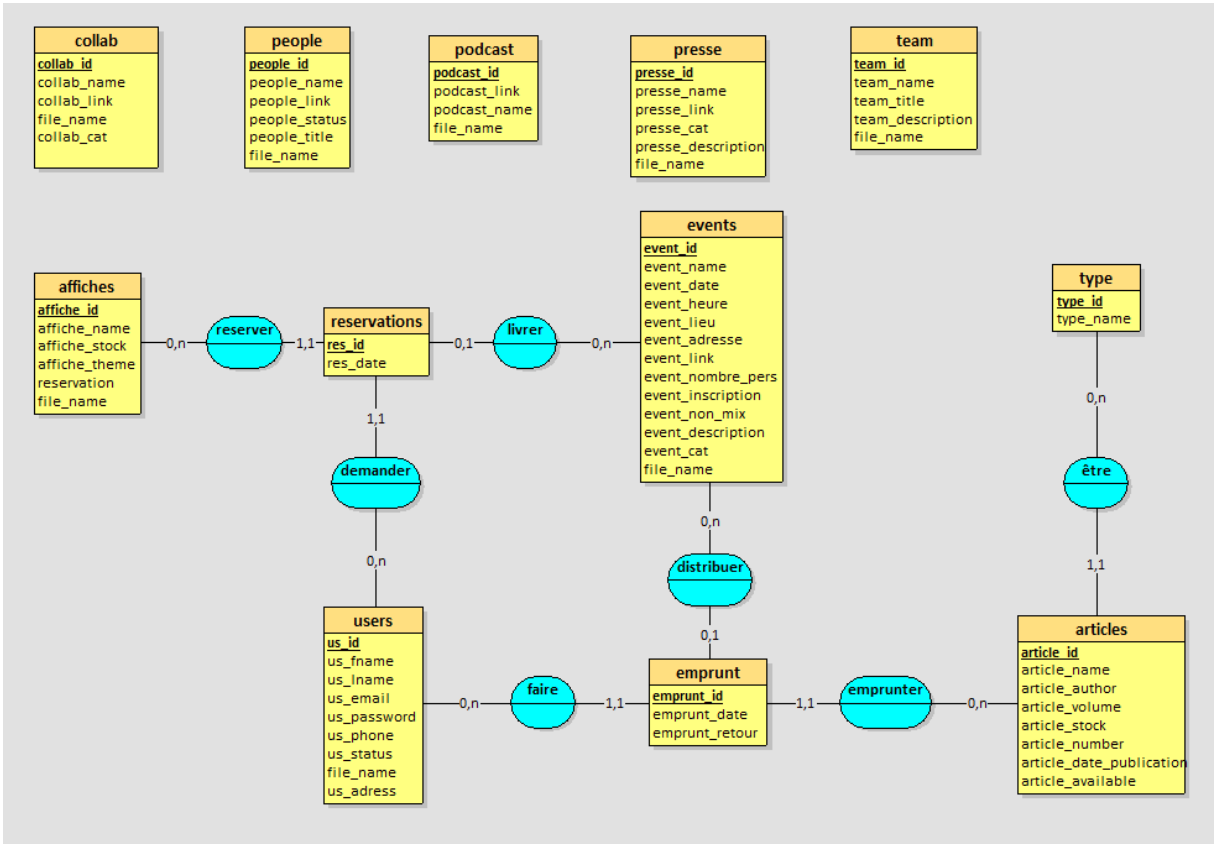
b. Diagramme d’activité



## c. Diagramme des classes



## 2. MCD


collab
collab_id
collab_name
collab_link
file_name
collab_cat
people
people_id
people_name
people_link
people_status
people_title
file_name
podcast
podcast_id
podcast_link
podcast_name
file_name
presse
presse_id
presse_name
presse_link
presse_cat
presse_description
file_name
team
team_id
team_name
team_title
team_description
file_name
events
event_id
event_name
event_date
event_heure
event_lieu
event_adresse
event_link
event_nombre_pers
event_inscription
event_non_mix
event_description
event_cat
file_name
type
type_id
type_name
affiches
affiche_id
affiche_name
affiche_stock
affiche_theme
reservation
file_name
reserver
reservations
res_id
res_date
livrer
demander
être
distribuer
users
us_id
us_fname
us_lname
us_email
us_password
us_phone
us_status
file_name
us_adress
faire
emprunt
emprunt_id
emprunt_date
emprunt_retour
emprunter
articles
article_id
article_name
article_author
article_volume
article_stock
article_number
article_date_publication
article_available
0,n
1,1
0,1

## 3. L’une des maquettes

# 4. Développement

## 1. Partie utilisateur

### 1. 1. Les affiches

CodeIgniter est un framework PHP de type MVC (Model, View, Controller).

L'utilisateur est redirigé à la page des affiches quand il se connecte à l'espace membre du site.

```php
if ($checkLogin) {
    $this->session->set_userdata('isUserLoggedIn', TRUE);
    $this->session->set_userdata('userId', $checkLogin['us_id']);
    redirect( uri: 'users/posters/' . $lang);
} else {
```

```php
/**
 * show posters
 */

public function posters()
{
    $lang = $this->uri->segment(3);
    $data = array();
    $this->lang->load('content', $lang == '' ? 'fr' : $lang);
    $data['controller'] = 'users';
    $data['function'] = 'posters';

    if ($this->isUserLoggedIn) {
        $con = array(
            'us_id' => $this->session->userdata('userId')
        );
        $data = $this->accountData($con, $data['function']);
        // get all posters
        $data['posters'] = $this->Poster_Model->getAll();
        $data['account_poster_ad'] = $this->lang->line('account_poster_ad');
        $data['account_poster_booking'] = $this->lang->line('account_poster_booking');
        $data['account_navbar'] = 'posters';
        // Pass the user data and load view
        $this->load->view( view: 'layouts/header', $data);
        $this->load->view( view: 'users/navbar', $data);
        $this->load->view( view: 'users/poster', $data);
        $this->load->view( view: 'layouts/footer', $data);
    } else {
        redirect( uri: 'users/login');
    }
} // posters ends here
```

Le contrôleur va d'abord définir la langue dans laquelle il faudrait charger les données. Ensuite vérifier si l'utilisateur est connecté. Puis charger les données permettant d'afficher l'espace membre et les données des affiches en faisant appel au model. Et enfin transmettre les données aux vues et les afficher.

La méthode du modèle permettant de faire la requête à la base des données :

```php
/**
 * return all posters
 */
public function getAll()
{
    return $this->db->get('poster')->result();
}
```

Une partie de la vue fait une boucle sur le tableau contenant les affiches :

```php
<div class="row d-flex justify-content-center ">
    <?php foreach ($posters as $poster) { ?>
    <div class="col-sm-5 mt-3 mx-1">
        <div class="row">
            <img src="<?php echo base_url(); ?>assets\img\affiches\<?php echo $poster->poster_id ?>.png"  class="img-fluid" alt="<?php echo $poster->poster_name ?>" >
        </div>
        <div class="row d-flex justify-content-center">
            <a class="btn btn-primary bg-info" href="<?php echo base_url().'index.php/users/book/'.$poster->poster_id.'/'.$lang; ?>" role="button"><?php echo $account_poster_booking ?></a>
        </div>

    </div>
    <?php } ?>
</div>
```

Le rendu :

Si l’utilisateur clique sur « réserver » les détails de l’affiche et les modalités de réservation vont apparaitre. Il peut sélectionner parmi les événements à venir celui où il pourrait récupérer l’affiche.


Cette fois-ci on fait recours au modèle pour récupérer les données des événements à venir, l’identifiant de l’affiche étant transmise au contrôleur par la méthode Get comme on peut le voir sur le bout du code de la vue des affiches.

Si l’utilisateur valide la réservation, on sauvegarde dans la table réservation son identifiant, l’identifiant de l’affiche, l’identifiant de l’événement et la date de la

réservation.

```php
public function reservation()
{
    $lang = $this->input->post('lang');
    $this->lang->load('content', $lang == '' ? 'fr' : $lang);
    $data['controller'] = 'users';
    $data['function'] = 'reservation';

    if ($this->input->post()) {
        $this->form_validation->set_rules('us_id', $data['us_id'] = $this->lang->line('us_id'), 'required');
        $this->form_validation->set_rules('poster_id', $data['poster_id'] = $this->lang->line('poster_id'), 'required');
        $this->form_validation->set_rules('event', $data['event'] = $this->lang->line('event_id'), 'required');

        if ($this->form_validation->run() == true) {
            $date = new DateTime;
            $format = $date->format( format: 'Y-m-d');
            $reservation = array(
                'res_date' => $format,
                'us_id' => $this->input->post('us_id',true),
                'event_id' => $this->input->post('event', true),
                'poster_id' => $this->input->post('poster_id',true)
            );
            if ($this->Reservation_Model->new($reservation)) {
                $this->session->set_flashdata('poster_booked', $this->lang->line('poster_booked'));
                redirect( uri: 'users/posters');
            } else {
                $this->session->set_flashdata('poster_not_booked', $this->lang->line('poster_not_booked'));
                redirect( uri: 'users/posters');
            }
        } else {
            $this->session->set_flashdata('poster_not_booked', $this->lang->line('poster_not_booked'));
            redirect( uri: 'users/posters');
        }
    } else {
        $this->session->set_flashdata('poster_not_booked', $this->lang->line('poster_not_booked'));
        redirect( uri: 'users/posters');
    }
}
```

Le modèle pour ajouter la réservation :

```php
/**
 * insert new
 */
public function new($data)
{
    return (bool)$this->db->insert('reservation', $data);
}
```

On a pensé à la sécurité de l'application dans la mesure où l'utilisateur ne fait pratiquement aucune saisie des données. On contrôle quand même les données et on fait recours aux facilités offertes par le framework comme l'échappement automatique des entrées et la protection contre le cross-site scripting. En plus, le site utilise le protocole https.

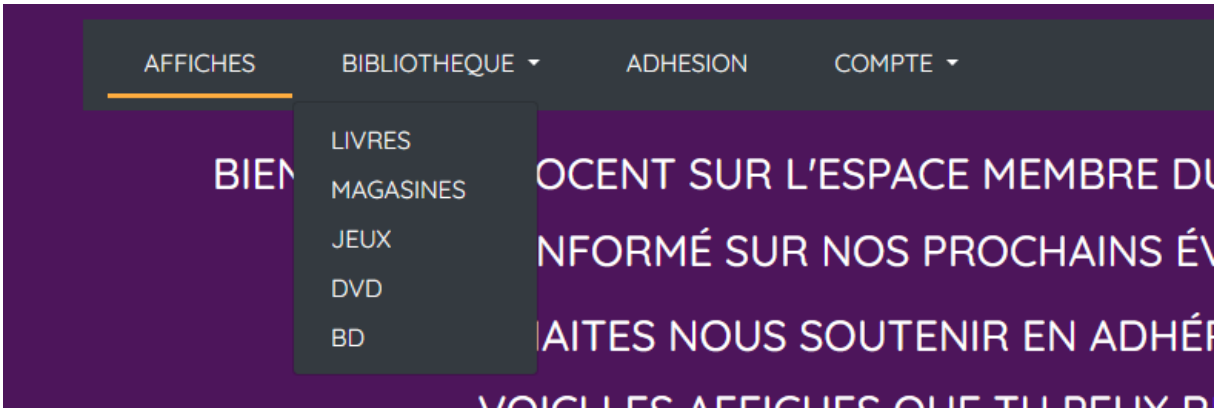
Et quand la réservation aboutit, on redirige l'utilisateur à la page d'accueil avec une message de remerciement.

1. 2. La bibliothèque

Au lieu de créer de multiples tables en fonction de type de documents (livres, magasines, DVD, BD, jeux) j'ai opté pour une seule table « articles » jointe à une autre « type » stockant seulement les types.

| # | Nom | Type de données | Taille/Ensem... | Non si... | NULL a... | ZERO... | Par défaut | Commentaire | Classement |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **article_id** | **INT** | **11** | ☐ | ☐ | ☐ | AUTO_INCREME... | | |
| 2 | article_name | VARCHAR | 200 | ☐ | ☑ | ☐ | NULL | | utf8_general_ci |
| 3 | article_author | VARCHAR | 100 | ☐ | ☑ | ☐ | NULL | | utf8_general_ci |
| 4 | article_volume | INT | 11 | ☐ | ☑ | ☐ | NULL | | |
| 5 | article_stock | INT | 11 | ☐ | ☑ | ☐ | NULL | | |
| 6 | article_number | VARCHAR | 50 | ☐ | ☑ | ☐ | NULL | | utf8_general_ci |
| 7 | article_date_p... | DATE | | ☐ | ☑ | ☐ | NULL | | |
| 8 | article_available | TINYINT | 4 | ☐ | ☑ | ☐ | NULL | | |
| 9 | type_id | INT | 11 | ☐ | ☐ | ☐ | Pas de défaut | | |
| 10 | article_state | VARCHAR | 50 | ☐ | ☑ | ☐ | NULL | | utf8_general_ci |
| 11 | article_comm... | VARCHAR | 100 | ☐ | ☑ | ☐ | NULL | | utf8_general_ci |
| 12 | file_name | VARCHAR | 50 | ☐ | ☑ | ☐ | NULL | | utf8_general_ci |

L'utilisateur accède aux articles après la connexion à l'espace membre du site. Une barre de navigation lui permet de sélectionner le type de documents qu'il veut emprunter.




On reste dans le schéma MVC. Une méthode du contrôleur permet de gérer les actions des utilisateurs sur le champ déroulant de la bibliothèque. Et deux autres, les emprunts et leur enregistrement.

Une partie du contrôleur :

```php
if ($this->isUserLoggedIn) {
    $con = array(
        'us_id' => $this->session->userdata('userId')
    );
    $data = $this->accountData($con, $data['function']);
    // get articles
    $this->load->model( model: 'Articles_Model');
    $data['articles'] = $this->Articles_Model->getArticles($type);
    // difining some specific data depending on type of articles
    switch ($type) {
        //books
        case '1':
            $data['book_title'] = $this->lang->line('book_title');
            $data['book_author'] = $this->lang->line('book_author');
            $viewName = "books";
            break;
        //magazines
        case '4':
            $data['magasine_name'] = $this->lang->line('magasine_name');
            $data['magasine_numero'] = $this->lang->line('magasine_numero');
            $viewName = "magasines";
            break;
        //DVD, games and bd
        case '2':
        case '3':
        case '5':
            $viewName = "games_dvd_bd";
            break;
    }
    $data['account_navbar'] = 'library';
    $data['borrow'] = $this->lang->line('borrow');
    // Pass the user data and load view
    $this->load->view( view: 'layouts/header', $data);
    $this->load->view( view: 'users/navbar', $data);
    $this->load->view( view: 'users/' . $viewName, $data);
    $this->load->view( view: 'layouts/footer', $data);
```

Le model :

```php
/**
 * get articles  by type
 */
public function getArticles($id)
{
    return $this->db->get_where('articles', array('type_id' => $id))->result();
}
```

Les vues diffèrent selon le type de document. Les livres et les magazines sont dans un tableau. Les jeux, dvd et bd dans un flex box.

Livres :

```php
<table class="table table-striped table-dark">
    <thead>
    <tr>
        <th scope="col"><?php echo $book_title?></th>
        <th scope="col"><?php echo $book_author?></th>
        <th scope="col"> </th>
    </tr>
    </thead>
    <tbody>
    <?php foreach ($articles as $book) {?>
    <tr>
        <td><?php echo $book->article_name ?></td>
        <td><?php echo $book->article_author ?></td>
        <td><a href="<?php echo base_url().'index.php/users/borrow/'.$book->article_id.'/'.$lang; ?>" class="text-warning"><?php echo $borrow ?></a></td>
    </tr>
    <?php } ?>
    </tbody>
</table>
```

Le rendu :


Les magazines :

```php
<table class="table table-striped table-dark">
    <thead>
    <tr>
        <th scope="col"><?php echo $magasine_name?></th>
        <th scope="col"><?php echo $magasine_numero?></th>
        <th scope="col"></th>
    </tr>
    </thead>
    <tbody>
    <?php foreach ($articles as $magasine) {?>
        <tr>
            <td><?php echo $magasine->article_name ?></td>
            <td><?php echo $magasine->article_number ?></td>
            <td ><a href="<?php echo base_url().'index.php/users/borrow/'.$magasine->article_id.'/'.$lang; ?>" class="text-warning"><?php echo $borrow ?></a></td>
        </tr>
    <?php } ?>
    </tbody>
</table>
```

Le rendu :


Les jeux, dvd et bd :

```php
<div class="row d-flex justify-content-center ">
    <?php foreach ($articles as $game) { ?>
        <div class="col-sm-3 mt-3 mx-1">
            <p><?php echo $game->article_name?></p>
            <div class="row">
                <img src="<?php echo base_url(); ?>assets\img\games\game.png"  class="img-fluid" alt="<?php echo $game->article_name ?>" >
            </div>
            <div class="row d-flex justify-content-center">
                <a class="btn btn-primary text-warning" href="<?php echo base_url().'index.php/users/borrow/'.$game->article_id.'/'.$lang; ?>" role="button"><?php echo $borrow ?></a>
            </div>

        </div>
    <?php } ?>
</div>
```

Le rendu :


Comme les réservations, quand l'utilisateur clique sur « emprunter », il est redirigé sur une page qui reprend les informations de l'article et lui donne la possibilité de choisir l'événement où il pourrait récupérer l'article. Ensuite on sauvegarde dans la base des données.

Une partie du code du contrôleur permettant de prendre en compte les emprunts :

```php
if ($this->input->post()) {
    $this->form_validation->set_rules('us_id', '', 'required');
    $this->form_validation->set_rules('article_id', '', 'required');
    $this->form_validation->set_rules('event', '', 'required');
    $this->form_validation->set_rules('type', '', 'required');

    if ($this->form_validation->run() == true) {
        $date = new DateTime;
        $format = $date->format('Y-m-d');

        $borrowing = array(
            'borrowing_date' => $format,
            'us_id' => $this->input->post('us_id', true),
            'event_id' => $this->input->post('event', true),
            'article_id' => $this->input->post('article_id', true)
        );
        $this->load->model('Borrowing_Model');
        if ($this->Borrowing_Model->new($borrowing)) {
            $this->session->set_flashdata('borrowing_success', $this->lang->line('borrowing_success'));
            redirect('users/article/' . $type . '/' . $lang);
        } else {
            $this->session->set_flashdata('borrowing_fail', $this->lang->line('borrowing_fail'));
            redirect('users/article/' . $type . '/' . $lang);
        }
    } else {
        $this->session->set_flashdata('borrowing_fail', $this->lang->line('borrowing_fail'));
        redirect('users/article/' . $type . '/' . $lang);
    }
```

Le modèle :

```php
/**
 * insert new
 */
public function new($data)
{
    return (bool)$this->db->insert('borrowing', $data);
}
```

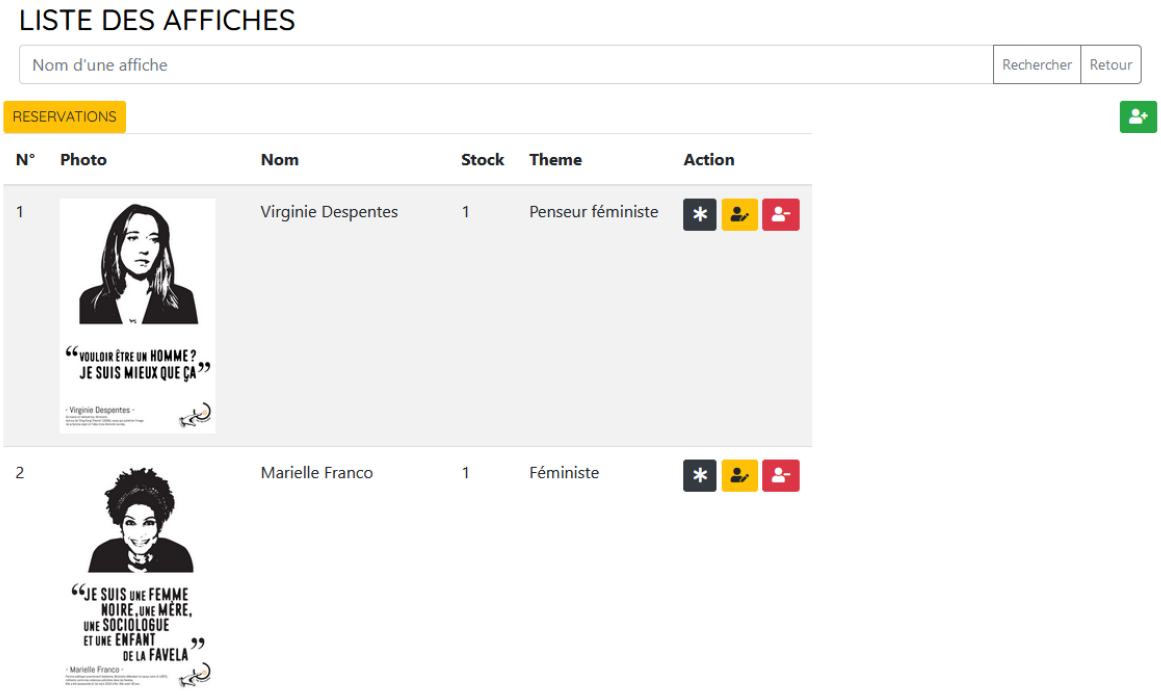
Niveau sécurité on garde l'échappement automatique et la protection contre cross-site scripting proposés par le framework.

## 2. Partie administrateur

### 2.1. Affiches

#### a. La liste des affiches

La partie admin du site permet de voir la liste des affiches, de faire la recherche d'une affiche, d'ajouter, voir les détails, modifier et supprimer une affiche. Bref tout le CRUD et un bouton permettant de voir les réservations.



Une partie du code du contrôleur :

```php
    //if search request is submitted
    if ($this->input->post('submitSearch')) {
        $inputKeywords = $this->input->post('searchKeywordAdmin');
        $searchKeywordAdmin = strip_tags($inputKeywords);
        if (!empty($searchKeywordAdmin)) {
            $this->session->set_userdata('searchKeywordAdmin', $searchKeywordAdmin);
        } else {
            $this->session->unset_userdata('searchKeywordAdmin');
        }
    } elseif ($this->input->post('submitSearchReset')) {
        $this->session->unset_userdata('searchKeywordAdmin');
    }
    $data['searchKeywordAdmin'] = $this->session->userdata('searchKeywordAdmin');
    //Get Rows count
    $conditions['searchKeywordAdmin'] = $data['searchKeywordAdmin'];
    $conditions['returnType'] = 'count';
    $rowCounts = $this->Poster_Model->getRowsPoster($conditions);
    //Pagination config
    $config['base_url'] = base_url() . 'index.php/posters/ad_index/';
    $config['uri_segment'] = 3;
    $config['total_rows'] = $rowCounts;
    $config['per_page'] = $this->perPage;
    //Initialize pagination library
    $this->pagination->initialize($config);
    //Define offset
    $page = $this->uri->segment(3);
    $offset = !$page ? 0 : $page;
    //Get rows
    $conditions['returnType'] = '';
    $conditions['start'] = $offset;
    $conditions['limit'] = $this->perPage;
    $data['posters'] = $this->Poster_Model->getRowsPoster($conditions);
    $data['ma_pages'] = 'ad_index_posters';
    $data['controller'] = 'posters';
    $data['function'] = 'ad_index';
    // Load the list page view
    $this->load->view('layouts/adheader', $data);
    $this->load->view('ad_posters/ad_index', $data);
    $this->load->view('layouts/adfooter');
```

On utilise la librairie de pagination de CodeIgniter.

Le modèle :

```php
/**
  * Returns rows from the database based on the conditions
  * @param array filter data based on the passed parameters
  */
public function getRowsPoster($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('poster_name' => $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("poster_id", $params)){
            $this->db->where('poster_id', $params['poster_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;
}//getRowsPoster ends here
```

La méthode ramène différents résultats en fonction des paramètres.

Dans la vue les affiches sont dans un tableau, avec une colonne contenant les liens pour voir les détails, modifier et effacer. Le lien pour ajouter est en haut à droite.

```php
<tr>
    <td><?php echo ucfirst($row['poster_id']); ?></td>
    <td><img src="<?php echo base_url(); ?>uploads/poster_img/<?php echo $row['file_name'] ?>"  width="150" class="img-fluid thumbnail" alt="<?php echo $row['poster_name'] ?>" ></td>
    <td><?php echo ucfirst($row['poster_name']); ?></td>
    <td><?php echo $row['poster_stock']; ?> </td>
    <td><?php echo ucfirst($row['poster_theme']); ?></td>
    <td>
        <a href="<?php echo base_url().'index.php/posters/ad_view_/'.$row['poster_id']; ?>" class="btn btn-sm btn-primary"><i class="fas fa-asterisk"></i></a>
        <a href="<?php echo base_url().'index.php/posters/edit_poster/'.$row['poster_id']; ?>" class="btn btn-sm btn-warning"><i class="fas fa-user-edit"></i></a>
        <a href="<?php echo base_url().'index.php/posters/delete_poster/'.$row['poster_id']; ?>" class="btn btn-sm btn-danger" onclick="return confirm('Voulez vous vraiment supprimer
    </td>
</tr>
```

### b. L’ajout d’une affiche

C’est un formulaire permettant de saisir le nom, le thème et le nombre d’exemplaire d’une affiche et de charger une photo.

Une partie du code du contrôleur :

```php
if ($this->input->post('poster_add')) {
    // Form field validation rules
    $this->form_validation->set_rules('poster_name', 'nom', 'required');
    $this->form_validation->set_rules('poster_theme', 'theme', 'required');
    $this->form_validation->set_rules('poster_stock', 'stock', 'required');
    $this->form_validation->set_rules('image', 'image', 'callback_file_check');
    // Prepare gallery data
    $formArray = array(
        'poster_name' => $this->input->post('poster_name', TRUE),
        'poster_theme' => $this->input->post('poster_theme', TRUE),
        'poster_stock' => $this->input->post('poster_stock', TRUE)
    );
    // Validate submitted form data
    if ($this->form_validation->run() == true) {
        // Upload image file to the server
        if (!empty($_FILES['image'])) {
            $insert = $this->Poster_Model->insert($formArray);
            if ($insert) {
                $id = $this->db->insert_id();
                $file = explode(".", $_FILES["image"]["name"]);
                $imageExtension = end($file);
                $imageName = $id . $imageExtension;
                // File upload configuration
                $config['upload_path'] = $this->uploadPath;
                $config['allowed_types'] = 'jpg|jpeg|png|gif';
                $config['file_name'] = $imageName;
                // 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();
                    $Array['file_name'] = $fileData['file_name'];
                } else {
                    $error = $this->upload->display_errors();
                }
            }
```

Le modèle :

```php
/**
* Insert poster data into the database
* @param $data data to be insert based on the passed parameters
*/
public function insert($data = array()) {
    if(!empty($data)){
        // Insert poster data
        $insert = $this->db->insert($this->table, $data);

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

La vue :

```php
<div class="col-9 container-fluid">
    <?php echo form_open_multipart() ;?>
    <div class="form-group">
        <label>Nom : </label>
        <input type="text" name="poster_name" class="form-control" value="<?php echo !empty($poster['poster_name'])?$poster['poster_name']:''; ?>" >
        <?php echo form_error('poster_name','<p class="help-block text-danger">','</p>'); ?>
    </div>
    <div class="form-group">
        <label>Theme : </label>
        <input type="text" name="poster_theme" class="form-control" value="<?php echo !empty($poster['poster_theme'])?$poster['poster_theme']:''; ?>" >
        <?php echo form_error('poster_theme','<p class="help-block text-danger">','</p>'); ?>
    </div>
    <div class="form-group">
        <label>Stock : </label>
        <input type="text" name="poster_stock" class="form-control" value="<?php echo !empty($poster['poster_stock'])?$poster['poster_stock']:''; ?>" >
        <?php echo form_error('poster_stock','<p class="help-block text-danger">','</p>'); ?>
    </div>
    <div class="form-group">
        <label>Images (.jpg .jpeg .png .gif) :</label>
        <input type="file" name="image" class="form-control" value="<?php echo !empty($poster['file_name'])?$poster['file_name']:''; ?>">
        <?php echo form_error('image','<p class="help-block text-danger">','</p>'); ?>
    </div>
    <div class="form-group">
        <a href="<?php echo base_url().'index.php/posters/ad_index'; ?>" class="btn btn-sm btn-secondary"><i class="fas fa-backspace"></i></a>
        <input type="submit" name="poster_add" class="btn btn-success btn-sm" value="Ajouter">
    </div>
    <?php echo form_close(); ?>
```

Le rendu :


AJOUT D'UNE AFFICHE
Nom :
Theme :
Stock :
Images (.jpg .jpeg .png .gif) :
Parcourir...
Aucun fichier sélectionné.
Ajouter

### c. Détails

Le lien permet de voir l'affiche en intégralité et des liens pour la modifier ou supprimer et pour un retour sur la liste des affiches.

Le contrôleur :

```php
public function ad_view_($id)
{
    if ($this->isUserLoggedIn) {
        $data = array();
        // Check whether id is not empty
        if (!empty($id)) {
            $con = array('poster_id' => $id);
            $data['poster'] = $this->Poster_Model->getRowsPoster($con);

            $data['controller'] = 'poster';
            $data['ma_pages'] = $data['function'] = 'ad_view_poster';
        } else {
            redirect('posters/ad_index');
        }
        // Load the add page view
        $this->load->view('layouts/adheader', $data);
        $this->load->view('ad_posters/ad_view', $data);
        $this->load->view('layouts/adfooter');
    } else {
        redirect('users/login');
    }
} // ad_view_poster ends here
```

La méthode pour le modèle est la même que celle utilisée pour la liste. Elle ramène les données d'une seule affiche quand l'identifiant d'une affiche est défini.

### d. Modification

Quand on clique sur le bouton de modification, un formulaire s'affiche avec les données de l'affiche pré remplies dans le champ de saisie.

Une partie du code du contrôleur :

```php
    if (empty($error)) {
        // Update image data
        $update = $this->Poster_Model->update($formArray, $id);
        if ($update) {
            $this->session->set_userdata('success_msg', 'Mise à jour - Réussie.');
            redirect('posters/ad_index');
        } else {
            $error = 'Quelques problèmes sont survenus, veuillez réessayer.';
        }
    }
```

Le modèle :

```php
public function update($data, $id) {

    if(!empty($data) && !empty($id)){

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

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

La vue est un formulaire :

```php
<?php echo form_open_multipart() ;?>
<div class="form-group">
    <label>Nom : </label>
    <input type="text" name="poster_name" class="form-control" value="<?php echo !empty($poster['poster_name'])?$poster['poster_name']:''; ?>" >
    <?php echo form_error('poster_name','<p class="help-block text-danger">','</p>'); ?>
</div>


<div class="form-group">
    <label>Theme : </label>
    <input type="text" name="poster_theme" class="form-control" value="<?php echo !empty($poster['poster_theme'])?$poster['poster_theme']:''; ?>" >
    <?php echo form_error('poster_theme','<p class="help-block text-danger">','</p>'); ?>
</div>

<div class="form-group">
    <label>Stock : </label>
    <input type="text" name="poster_stock" class="form-control" value="<?php echo !empty($poster['poster_stock'])?$poster['poster_stock']:''; ?>" >
    <?php echo form_error('poster_stock','<p class="help-block text-danger">','</p>'); ?>
</div>
```

Le rendu :

# MODIFICATION D'UNE AFFICHE

Nom :

Andrea Workin

Theme :

Witches week

Stock :

5

Images (.jpg .jpeg .png .gif) :

Parcourir... Aucun fichier sélectionné.


Modifier

### e. Suppression

Après une question de confirmation, le lien supprime une affiche dans la base de données.

Le contrôleur :

```php
public function delete_poster($id)
{
    if ($this->isUserLoggedIn) {
        // Check whether id is not empty
        if ($id) {
            $con = array('poster_id' => $id);
            $formArray = $this->Poster_Model->getRowsPoster($con);
            // Delete poster data
            $delete = $this->Poster_Model->delete($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('posters/ad_index');
            } else {
                $this->session->set_userdata('error_msg', 'Veuillez rééssayer');
            }
        }
    } else {
        redirect('users/login');
    }
} // delete poster ends here
```

Le modèle :

```php
public function delete($id){

    $delete = $this->db->delete($this->table, array('poster_id' => $id));

    return $delete?true:false;

} //delete ends here
```

## 2.2. Réservations

La partie administrateur des réservations permet de voir une liste des réservations avec des détails, avec une barre de recherche pour trouver une réservation par utilisateur ou affiche ou événement, un lien pour voir l'affiche réservée, des boutons d'ajout, modification et suppression.

### a. La liste


Une partie du code du contrôleur :

```
//Get rows
$conditions['returnType'] = '';
$conditions['start'] = $offset;
$conditions['limit'] = $this->perPage;
$data['reservations'] = $this->Reservation_Model->getRowsReservation($conditions);

$data['ma_pages'] = 'ad_index_reservations';

$data['controller'] = 'reservation';
$data['function'] = 'ad_index';


// Load the list page view
$this->load->view('layouts/adheader', $data);
$this->load->view('ad_reservation/ad_index', $data);
$this->load->view('layouts/adfooter');
```

Le modèle :

```php
public function getRowsPoster($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('poster_name' => $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("poster_id", $params)){
            $this->db->where('poster_id', $params['poster_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;
}//getRowsPoster ends here
```

La vue :

```
<table class="table table-responsive table-boredered table-striped">
    <thead>
    <tr>
        <th>N°</th>
        <th>Date de reservation</th>
        <th>Utilisateur </th>
        <th>Affiche </th>
        <th>Evénement </th>
        <th>préparation </th>
        <th>Livraison </th>
        <th colspan="3">Action</th>
    </tr>
    </thead>
    <tbody>
    <?php if(!empty($reservations)){
        foreach($reservations as $row){
```

b. Ajout d'une réservation

Quand on clique sur le bouton vert, un formulaire permettant de sélectionner l'utilisateur, l'affiche et l'événement se charge. On renseigne aussi si l'affiche est préparée ou pas.


Le contrôleur :

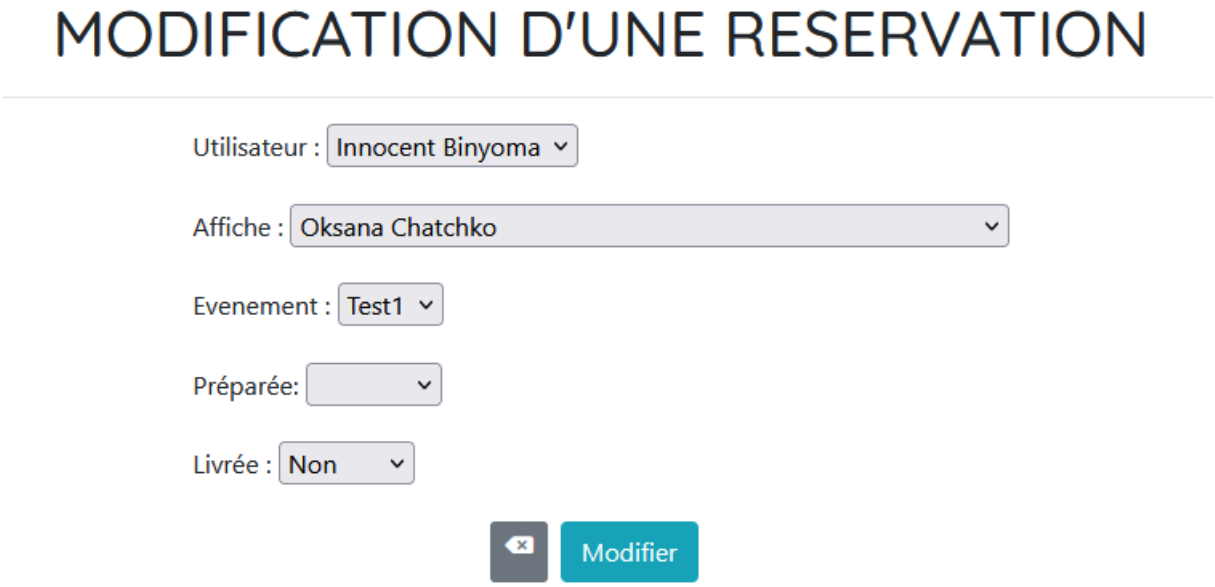
```php
$params = array();
$params['function']='add_reservation';
$params['view'] = 'add';

if ($this->input->post()) {
    $this->form_validation->set_rules('us_id', 'Utilisateur', 'required');
    $this->form_validation->set_rules('poster_id', 'Affiche', 'required');
    $this->form_validation->set_rules('event', 'Evenement', 'required');
    $this->form_validation->set_rules('res_preparation', 'préparation', 'required');

    if ($this->form_validation->run() == true) {
        $date = new DateTime;
        $format = $date->format('Y-m-d');
        $data = array(
            'res_date' => $format,
            'us_id' => $this->input->post('us_id', true),
            'event_id' => $this->input->post('event', true),
            'poster_id' => $this->input->post('poster_id', true),
            'res_preparation' => $this->input->post('res_preparation', true)
        );
        if ($this->Reservation_Model->new($data)) {
            $this->session->set_flashdata('success_msg', 'la reservation a été enregistrée');
            redirect('reservation/ad_index');
        } else {
            $this->session->set_flashdata('error_msg', ' Une erreur s\'est produite! réessayer');
            $this->loadData($params);
        }
    } else {
        $this->session->set_flashdata('error_msg', ' Une erreur s\'est produite! réessayer');
        $this->loadData($params);
    }
} else {
    $this->loadData($params);
```

Le modèle :

```php
/**
 * insert new
 */
public function new($data)
{
    return (bool)$this->db->insert('reservation', $data);
}
```

## c. Modification

Le bouton de modification renvoie à un formulaire pré rempli.




Le contrôleur :

```php
if ($this->form_validation->run() == true) {

    $reservation = array(
        'us_id' => $this->input->post('us_id', true ),
        'event_id' => $this->input->post('event', true),
        'poster_id' => $this->input->post('poster_id', true),
        'res_preparation' => $this->input->post('res_preparation', true),
        'res_livraison' => $this->input->post('res_livraison', true)
    );

    if ($this->Reservation_Model->update($reservation, $id)) {
        $this->session->set_flashdata('success_msg', 'la reservation a été enregistrée');
        redirect('reservation/ad_index');
    } else {
        $this->session->set_flashdata('error_msg', ' Une erreur s\'est produite! réessayer');
        $this->loadData($params);
```

Modèle :

```php
/**
 * @param $data
 * @param $id
 * @return bool
 */
public function update($data, $id)
{
      $this->db->where('res_id', $id);
      return (bool)$this->db->update('reservation',$data);
}
```

# CONCLUSION

La période en stage a été enrichissante du point de vue professionnelle et personnelle.

Du point de vue professionnelle, c'était une opportunité de mettre en pratique les connaissances acquises lors de ma formation et d'acquérir de nouvelles surtout vers la fin où il a été question de développer une application mobile avec flutter.

Du point de vue personnelle, c'était une rencontre avec des gens qui luttent contre les discriminations, ce qui rend le monde meilleur.