

# Documentation Technique

## Introduction

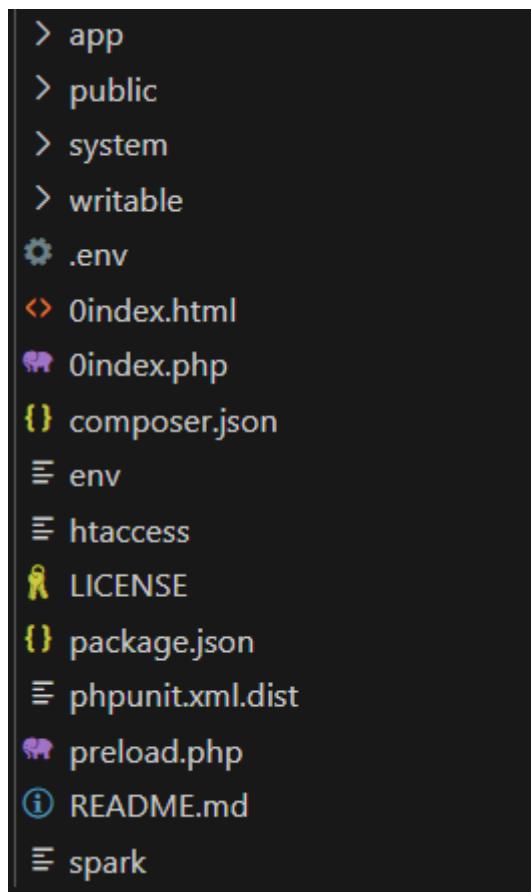
Cette documentation a pour but de lister le fonctionnement du site web dans le cadre de futurs projets de développement. Vous retrouverez dans cette documentation les fonctionnalités listées et détaillées, ainsi que leur fonctionnement.

## Table des matières

Introduction .....	1
Architecture Code Igniter 4 .....	2
Base de données .....	10
Accueil .....	10
Map .....	10
Header .....	11
Accueil .....	12
Liste Récit .....	14
Récit .....	17
Modification Récit .....	17
Suppression Récit .....	20
Statistique .....	21
Langue .....	21
Sidebar .....	22
Sélectionner un Type de Lieu .....	23
Sélectionner un Récit .....	25
Menu de Gestion .....	27
Déconnecté .....	28
Connexion .....	28
Ajout Point .....	30
Ajout Récit .....	30
Ajout Polygone .....	30
Ajout Esclave/Auteur .....	30
Modification d'un Esclave/auteur .....	31
Suppression d'un Esclave/auteur .....	31
Footer .....	31
Contact .....	31
Information .....	33

# Architecture Code Igniter 4

Code Igniter 4 utilise le modèle MVC (Modèle Vue Contrôleur).

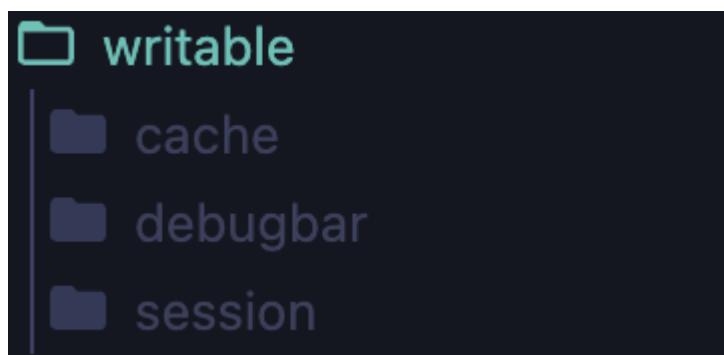


Dans le projet, on retrouve 5 parties :

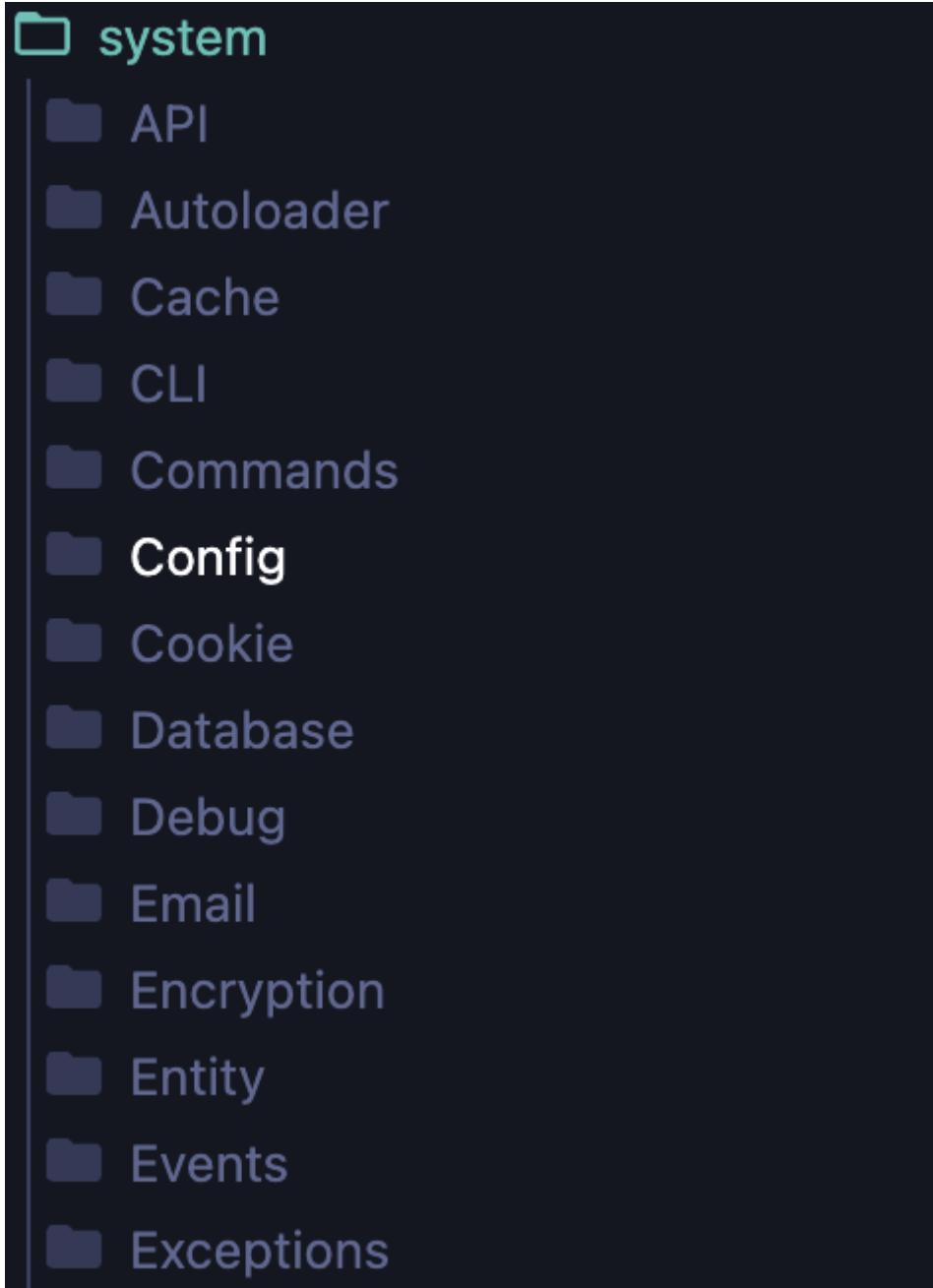
- La première est la configuration du projet Code Igniter avec le fichier .env à modifier pour le fonctionnement de votre site.



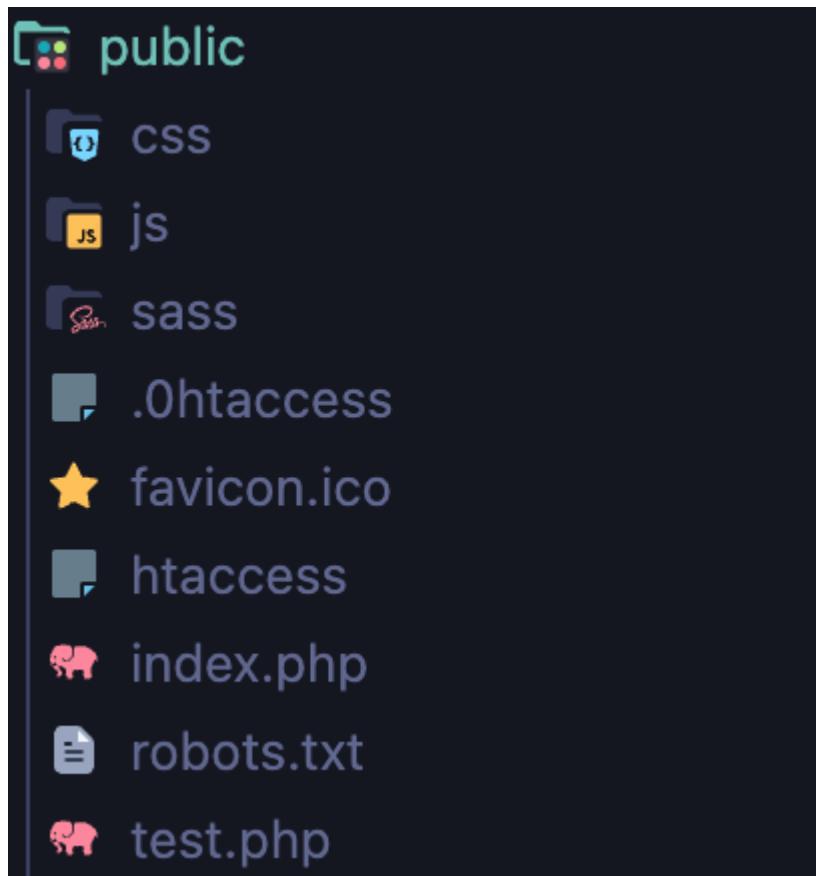
- La deuxième est les writables, ils stockent toutes les informations liées à des problèmes.



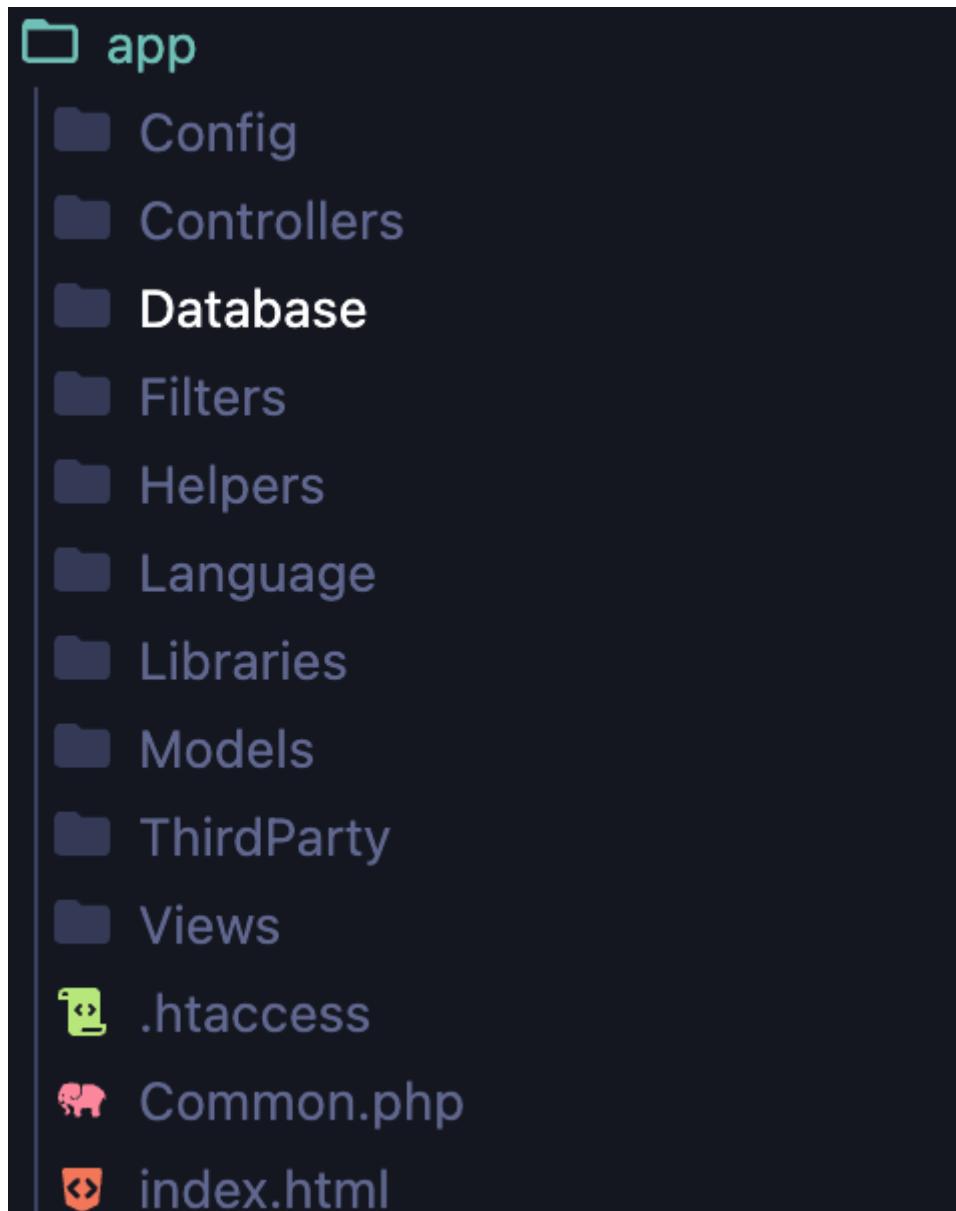
- Le troisième est le système. Tout le fonctionnement du site est contenu dedans.



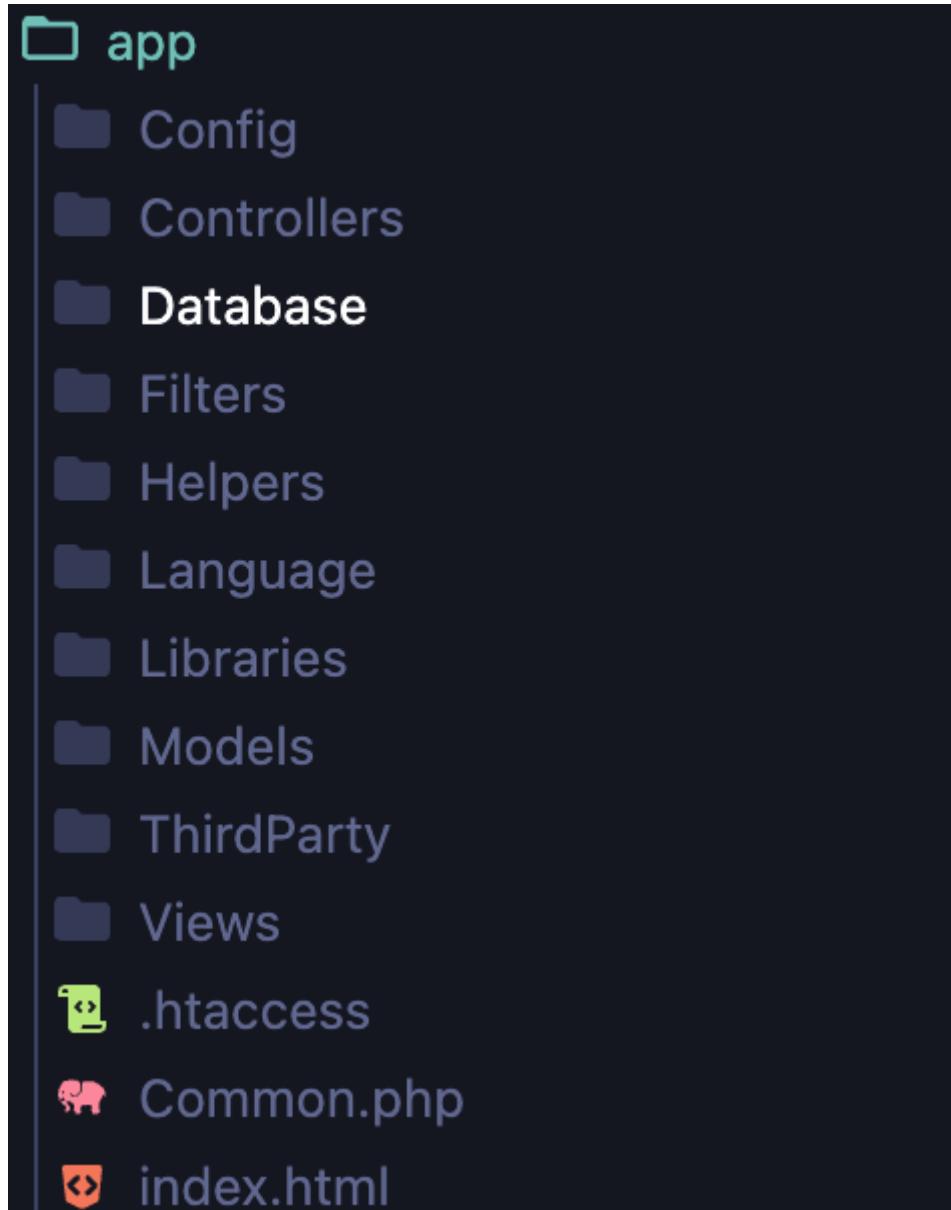
- Le quatrième est le dossier public. Dans celui-ci, on peut retrouver les fichiers css, js ou tout autre langage utilisé.



- Le dernier est app, dans celui-ci on retrouve toute l'application, les vues, les contrôleurs et les modèles.



Rentrerons dans app pour voir en détail l'application.



Dans app, on retrouve plusieurs dossiers, mais nous allons nous focaliser sur certains dossiers en particulier.

- Le dossier config, où vous pourrez paramétriser l'application et définir les routes de votre projet. L'application utilise énormément les routes. Grâce à elles, on peut se déplacer dans l'application et réaliser des actions.

## Config

Boot

App.php

Autoload.php

Cache.php

Constants.php

ContentSecurityPolicy.php

Cookie.php

Cookie0.php

CURLRequest.php

Database.php

DocTypes.php

Email.php

Encryption.php

Events.php

Exceptions.php

Feature.php

Filters.php

ForeignCharacters.php

Format.php

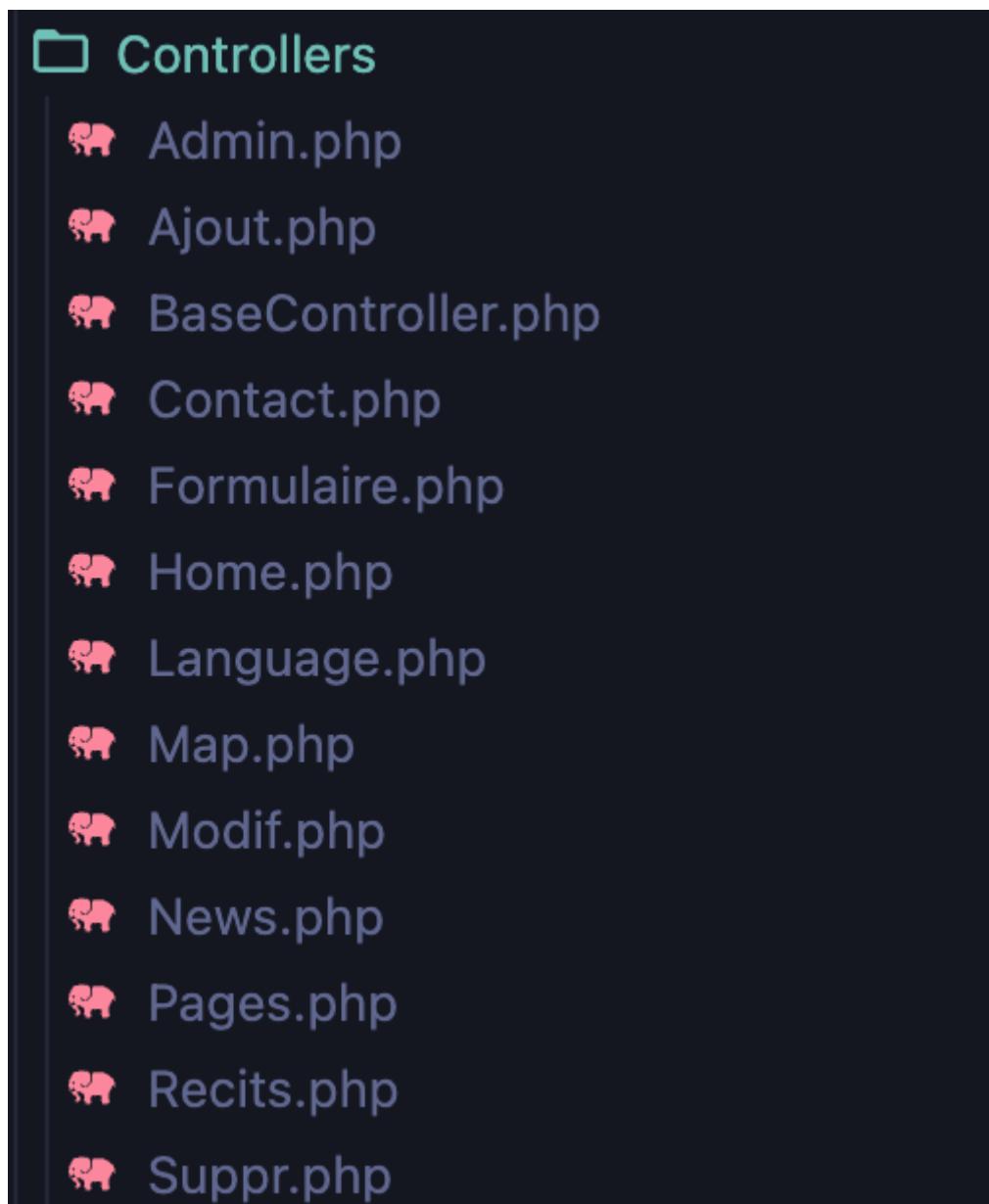
Generators.php

Honeypot.php

Images.php

Kint.php

- Le dossier contrôleur, où on trouve les contrôleurs qui permettent la gestion des vues. Les contrôleurs importent les méthodes qui sont définies dans les modèles.



- Le dossier modèle, où il y a tous les modèles de méthodes. On y retrouve des méthodes de requête en base de données ou de récupération de données qui seront appliquées dans les contrôleurs.

## Models

- .gitkeep
- ModelAdmin.php
- ModelCouches.php
- ModelFormulaire.php
- ModelMap.php
- ModelRecit.php
- NewsModel.php

- Le dossier langage, où on trouve la traduction de tous les champs affichés du site. Actuellement, il y a la version française et anglaise.

## Language

- en
- fr

- Le dossier views qui stocke toutes les pages du site web, rangées dans des dossiers. Le dossier templates contient les pages pour le footer et la sidebar. Le dossier reclaves, quant à lui, contient toutes les autres pages du site web ainsi que le header.

## Views

- errors
- news
- pages
- reclaves
- templates
- welcome\_message.php

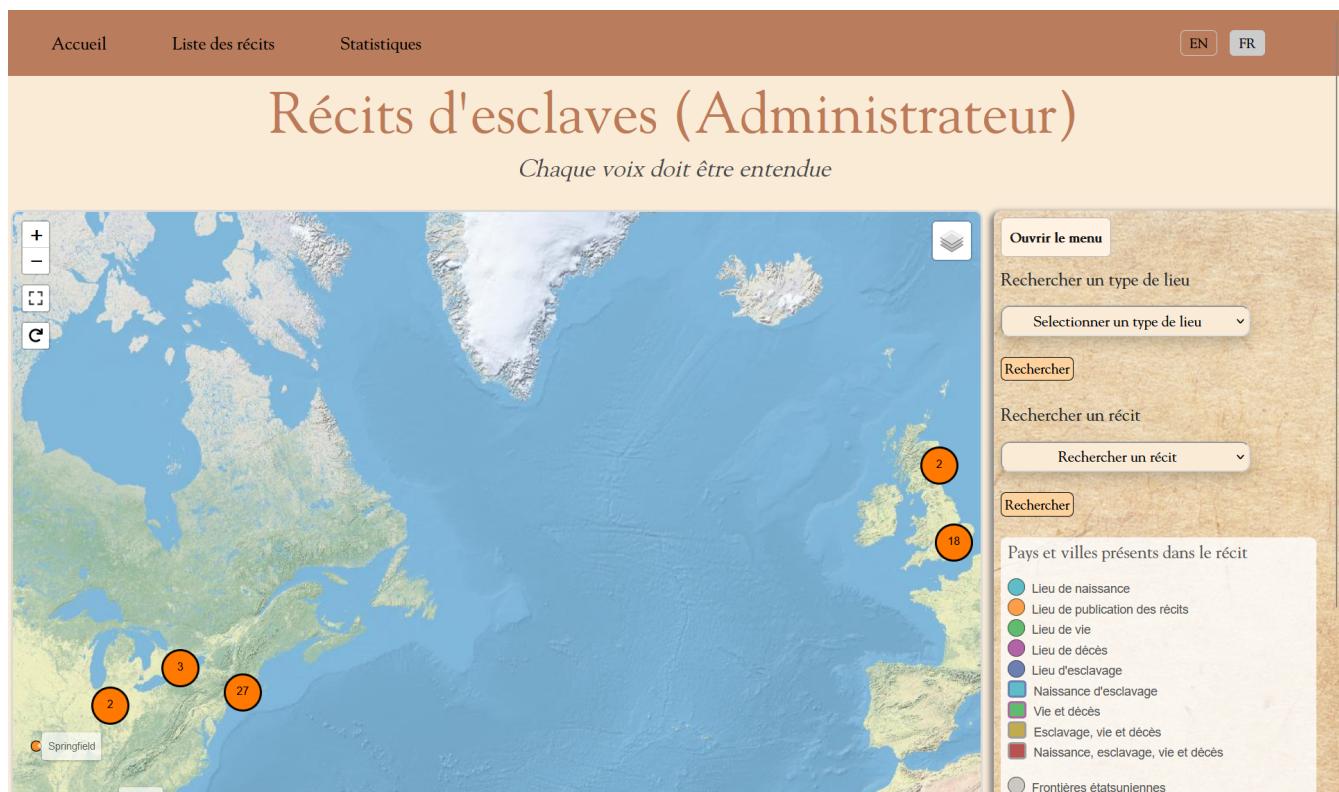
# Base de données

Maintenant que vous avez vu la structure du projet, passons aux fonctionnalités.

## Accueil

### Map

L'accueil est la page principale du projet, elle redirige vers la plupart des fonctionnalités du site web.



Accueil correspond à la page [accueil.php](#).

La carte correspond à :

```

var map = L.map('carte').setView([29.052497808641004, -45.60848140244032],3);
map.addControl(new L.Control.Fullscreen());

// Fond ESRI relief
var Esri_WorldShadedRelief = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Shaded_Relief/MapServer/tile/{z}/{y}/{x}', {
    attribution: 'Tiles © Esri — Source: Esri',
    maxZoom: 13
});

// Fond World Terrain Base
var ESRI_Terrain_Base = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Terrain_Base/Maps/MapServer/tile/{z}/{y}/{x}', {
    attribution: 'Tiles © Esri — Source: USGS, Esri, TANA, DeLorme, and NPS',
    maxZoom: 13
});

// Fond ESRI World Physical
var Esri_WorldPhysical = L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Physical_Map/MapServer/tile/{z}/{y}/{x}', {
    attribution: 'Tiles © Esri — Source: US National Park Service',
    maxZoom: 8
}).addTo(map);

var OpenTopoMap = L.tileLayer('https://tile.opentopomap.org/{z}/{x}/{y}.png', {
    maxZoom: 17,
    attribution: 'Map data: © <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
});

var OpenStreetMap_Mapnik = L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19,
    attribution: '© <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
});

```

Voici le code JavaScript pour ajouter les cartes à la page. Ensuite, on retrouve l'ajout de boutons pour manipuler la map, la légende, et enfin l'ajout de cercles. L'image ne contient que la récupération des maps.

[Accueil.php](#) ne contient que la carte; la sidebar à droite vient de la page [sidebar.php](#).

## Header



Le fichier est [header\\_resc.php](#).

```

<nav class="navbar navbar-expand-lg ">
    <a class="navbar-brand" href="= site_url() . "map" ?&gt;&lt;?= lang('headergeo.nav_bar.home')?&gt;&lt;/a&gt;
    &lt;a class="navbar-brand" href="<?= site_url() . "recits" ?&gt;&lt;?= lang('headergeo.nav_bar.list_narratives')?&gt;&lt;/a&gt;

    &lt;?php if ($session-&gt;get('is_admin')) : ?&gt;
    &lt;a class="navbar-brand" href="<?= site_url('statistiques') ?&gt;&lt;?= lang('headergeo.nav_bar.statistics') ?&gt;&lt;/a&gt;
    &lt;?php endif; ?&gt;

    &lt;div class="language"&gt;
        &lt;a href="#" id="changeLanguageEN" class="language-link&lt;?php echo ($session-&gt;get('locale') === 'en') ? ' language-link-active' : '' ?&gt;
        &lt;a href="#" id="changeLanguageFR" class="language-link&lt;?php echo ($session-&gt;get('locale') === 'fr') ? ' language-link-active' : '' ?&gt;
    &lt;/div&gt;
&lt;/nav&gt;
</pre

```

On y retrouve une barre de navigation ([navbar](#)) avec :

- **Accueil** qui renvoie vers la map en utilisant la route `map`
- La liste des récits avec la route `recit`
- Statistiques avec sa route.

De plus, on y retrouve le code pour définir la langue du site.

```
document.getElementById('changeLanguageEN').addEventListener('click', function(event) {
  event.preventDefault();

  var form = document.createElement('form');
  form.method = 'post';
  form.action = '<?php echo base_url('language/changeLanguage/en'); ?>';

  // Ajouter le champ pour l'ID
  var idInput = document.createElement('input');
  idInput.type = 'hidden';
  idInput.name = 'idE';
  idInput.value = '<?php echo isset($_POST['idE']) ? htmlspecialchars($_POST['idE']) : null; ?>'; // Remplacez par votre valeur
  form.appendChild(idInput);

  // Ajouter le champ pour l'ID
  var idInput = document.createElement('input');
  idInput.type = 'hidden';
  idInput.name = 'boutonaj';
  idInput.value = '<?php echo isset($_POST['boutonaj']) ? htmlspecialchars($_POST['boutonaj']) : null; ?>';
  form.appendChild(idInput);

  document.body.appendChild(form);
  form.submit();
});
```

Le script JavaScript est utilisé lorsque le bouton est cliqué, et il va faire une recherche dans le dossier `language`, ici en anglais.

## Accueil

Quand on clique sur `Accueil` dans le header, voici comment le code va exécuter cette action.

```
<a class="navbar-brand" href="= site_url() . "map" ?&gt;&gt;&lt;?= lang('headergeo.nav_bar.home')?&gt;&lt;/a&gt;</pre

```

Quand `Accueil` est cliqué, il va chercher la route `map`.

```
$routes->match(['get', 'post'], '/map', [Map::class, 'index']);
```

La route lui indique qu'il doit exécuter la méthode `index` de la classe `Map` (contrôleur Map).

```

public function index()
{
    $model = model(ModelMap::class);
    $model2 = model(ModelCouches::class);

    $model3 = model(ModelRoyAfr::class);
    $model4 = model(ModelAiresAut::class);

    $model5 = model(ModelPoints::class);
    $model6 = model(ModelPolygones::class);
    $model7 = model(ModelRecit_poly::class);
    helper('form');

    // affichage si l'utilisateur choisit le formulaire selon les récits
    if ($this->request->getPost('select_recit')) {

        $data2 = [
            'id_recit' => $this->request->getPost('select_recit')
        ];

        $list = $model7->search_recit_poly($data2);

        $data2 = [
            'points' => $model->getPoints(),
            'couche' => $model2->search_adv($data2),

            'aires' => $model4->getAiresAut(),
            'roy_afr' => $model3->getRoyAfr(),
            'pts' => $model5->search_pts($data2),
            'poly' => $model6->search_poly($list),
            'selec' => $this->request->getpost('select_recit')
        ];
    }
}

```

Il va importer les méthodes des modèles et les utiliser. Il teste si un des formulaires dans le sidebar est rempli, sinon il va faire l'affichage de base.

```

// affichage initial avec les différents récits
else {

    $data = [
        'points' => $model->getPoints(),
    ];
}

```

Il va exécuter la méthode `getPoints` du modèle `Map`.

```
protected $table = 'points';
protected $allowedFields = ['id', 'ville', 'id_recit','geoj', 'nom_esc', 'titre','date_publi'];

public function getPoints()
{
    return $this->asArray()
        ->join('tab_recits_v3', 'points.id_recit = tab_recits_v3.id_recit')
        ->where(['points.type'=> 'publication'])
        ->findAll(60);
}
```

Il va lier la table `tab_recit_v3` et `point` par leur `id_recit` et va retourner toutes les lignes dans `point` qui ont l'attribut `type` égal à `publication`. Les attributs des lignes récupérées par la requête seront égaux à la définition de `allowedFields` au-dessus de la méthode. Si un attribut n'est pas dans `allowedFields`, il ne sera pas récupéré.

```
DatabaseUtils::insertVisit('map_recits');
return view('resclaves/header')
    . view('resclaves/style')
    . view('templates/sidebar', $data2)
    . view('resclaves/accueil2', $data2)
    . view('templates/footer_resc');
```

Puis il va retourner les vues pour les afficher.

## Liste Récit

Lorsque l'on clique sur `Liste des récits`,

il va chercher la route `recit`.

```
<a class="navbar-brand" href="= site_url() . "recits" ?&gt;"&gt;<?= lang('headergeo.nav_bar.list_narratives')?&gt;&lt;/a&gt;</pre
```

La route lui indique qu'il doit utiliser la méthode `index` du contrôleur `Recits`.

```
$routes->match(['get', 'post'],'/recits', [Recits::class, 'index']);
```

La méthode effectue tout d'abord des requêtes pour connaître l'ordre de tri des récits.

```

public function index()
{
    $model = model(ModelRecit::class);

    $search = $this->request->getGet('search');
    $tri = $this->request->getGet('tri');
    $order = $this->request->getGet('tri');

    switch($tri){
        case "nomAZ":
            $tri = "nom_esc";
            $order = "ASC";
            break;
        case "nomZA":
            $tri = "nom_esc";
            $order = "DESC";
            break;
        case "anneeAZ":
            $tri = "date_publi";
            $order = "ASC";
            break;
        case "anneeZA":
            $tri = "date_publi";
            $order = "DESC";
            break;
        case "titreAZ":
            $tri = "titre";
            $order = "ASC";
            break;
        case "titreZA":
            $tri = "titre";
            $order = "DESC";
            break;
        default:
    }
}

```

Ensuite, elle effectue des requêtes pour rechercher les récits dans `tab_recit_v3`.

```

$data = [
    'recits' => $model->getRecits($search),
    'recitsT' => $model->getTriRecits($tri, $order)
];

if((!empty($search) && !empty($search)) || (!empty($tri) && !empty($tri))){
    DatabaseUtils::insertVisit('recits');
}

return view ('resclaves/header')
. view ('resclaves/recits',$data)
. view ('templates/footer_resc');
}

```

```

public function getRecits($search = null)
{
    if ($search) {
        return $this->asArray()
            ->like('nom_esc', $search) // Modifiez cette
            ->orLike('titre', $search)
            ->orLike('date_publi', $search)
            ->orderBy('date_publi')
            ->findAll();
    } else {
        return $this->asArray()
            ->orderBy('date_publi')
            ->findAll();
    }
}

public function getTriRecits($tri = null, $order = null)
{
    if ($tri) {
        return $this->asArray()
            ->orderBy($tri, $order)
            ->findAll();
    } else {
        return;
    }
}

```

Et enfin, elle retourne les vues pour les afficher ([recits.php](#)).

```

<tbody>
    <?php if(isset($recitsT) && is_array($recitsT)):
        | foreach ($recits as $r): ?>

<tr>
    <td>
        | <p><a href="= site_url()."recits/".$r['id_recit'], 'url' ?&gt;"&gt;&lt;?php echo $r['nom_esc'];?&gt;&lt;/a&gt;&lt;/p&gt;
    &lt;/td&gt;

    &lt;td&gt;
        | &lt;p&gt;&lt;?php echo $r['date_publi'];?&gt;&lt;/p&gt;
    &lt;/td&gt;

    &lt;td&gt;
        | &lt;p&gt;&lt;?php echo $r['titre'];?&gt;&lt;/p&gt;
    &lt;/td&gt;

        | &lt;?php if ($session-&gt;get('is_admin')) : ?&gt;
            |   &lt;td&gt;
                |     | &lt;p&gt;&lt;a href="<?= site_url('/modif_recit?esc='.$r['id_auteur']).'&amp;idR='.$r['id_recit']) ?&gt;"&gt;&lt;?= lang('recits.
            |   &lt;/td&gt;
            | 
            |   &lt;td&gt;
                |     | &lt;p&gt;&lt;a href="<?= site_url('Suppr/SupprRecit?esc='.$r['id_auteur']).'&amp;idR='.$r['id_recit']) ?&gt;" onclick="return confirm('
            |   &lt;/td&gt;
        | &lt;?php endif; ?&gt;

&lt;/tr&gt;
</pre

```

Le tableau affiche les récits, avec pour chaque ligne, un lien vers le récit en détail, ainsi que des possibilités de modification et de suppression des récits depuis la liste.

## Récit

### Modification Récit

```

<td>
    | <p><a href="= site_url('/modif_recit?esc='.$r['id_auteur']).'&amp;idR='.$r['id_recit']) ?&gt;"&gt;
&lt;/td&gt;
</pre

```

Chaque ligne `modifier` a pour lien `modif_recit` suivi des informations sur le récit sélectionné.

```
$routes->match(['get', 'post'], '/modif_recit', [Modif::class, 'modif']);
```

La route appelle la méthode `modif` du contrôleur `Modif`.

```

public function modif()
{
    $session = \Config\Services::session();
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);
    $model2 = model(ModelCouches::class);
    $model3 = model(ModelPolygones::class);
    $model4 = model(ModelRecit_poly::class);

    $data = [
        'title' => $model->getRecit(),
        'auteurs' => $model1->getAuteurs(),
        'polys' => $model3->getPoly(),
        'recitP' => $model4->getRecitPoly()
    ];

    $session = \Config\Services::session();

    if ($session->has('is_admin') && $session->get('is_admin') === true) {
        return view('resclaves/header')
            . view('resclaves/modif_recit', $data);
    } else {
        return redirect()->to('/map');
    }
}

```

La méthode récupère toutes les informations des récits, puis affiche la page de modification du récit avec un formulaire pour chaque champ.

```

<form action="= site_url('Modif/ModifPoly_Recit?idR=' . $_GET['idR']) ?&gt;" method="post"&gt;
    &lt;label&gt;<?= lang('modif_recit.name_narrative') ?&gt;&lt;/label&gt;
    &lt;?php
    if (!empty($title) &amp;&amp; is_array($title)) {
        foreach ($title as $elt) {
            if ($elt['id_recit'] == $_GET['idR']) {
                echo '&lt;input name="nomR" id="nomR" type="text" value="' . $elt['titre'] . '" required/&gt;' . br;
            }
        }
    }
    ?&gt;
    &lt;label&gt;<?= lang('modif_recit.name_slave') ?&gt;&lt;/label&gt;
    &lt;select name="idE" id="idE" required&gt;
        &lt;?php
        if (!empty($auteurs) &amp;&amp; is_array($auteurs)) {
            foreach ($auteurs as $elt) {
                if ($elt['id_auteur'] == $_GET['esc']) {
                    echo '&lt;option value="' . $elt['id_auteur'] . '" selected&gt;' . $elt['nom'] . ' &lt;/option&gt;';
                } else {
                    echo '&lt;option value="' . $elt['id_auteur'] . '"' . $elt['nom'] . ' </option>';
                }
            }
        }
        ?>
    </select><br><br>

```

Elle va remplir les champs en parcourant les résultats de la méthode lorsque l'id du récit est égal à l'id du récit venant de l'URL.

Une fois cela fait, dès que l'on valide le formulaire, celui-ci utilise la route `Modif/ModifPoly_Recit`.

```
$routes->post('Modif/ModifPoly_Recit', 'Modif::ModifPoly_Recit');
```

La route renvoie vers la méthode `ModifPoly_Recit` du contrôleur `Modif`.

```
public function ModifRecit()
{
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);
    $data = [
        'title' => $model->getRecit(),
        'auteurs' => $model1->getAuteurs()
    ];

    $nomR = $this->request->getPost('nomR');
    $idE = $this->request->getPost('idE');
    $idR = $this->request->getPost('idR');
    $lieuP = $this->request->getPost('lieuP');
    $infoSup = $this->request->getPost('infoSup');
    $dateP = $this->request->getPost('dateP');
    $typeR = $this->request->getPost('typeR');
    $com = $this->request->getPost('com');
    $modeP = $this->request->getPost('modeP');
    $nomS = $this->request->getPost('nomS');
    $lienR = $this->request->getPost('lienR');
    $nb = $this->request->getPost('nb');
    for($i=0; $i<$nb; $i++){
        $type[$i] = $this->request->getPost('type' . $i);
    }
    for($i=0; $i<$nb; $i++){
        $idP[$i] = $this->request->getPost('idP' . $i);
    }
    for($i=0; $i<$nb; $i++){
        $nomP[$i] = $this->request->getPost('nomP' . $i);
    }
}
```

La méthode va récupérer tous les champs du formulaire.

```

$nomE = '';
foreach ($data['auteurs'] as $elt) {
    if($elt['id_auteur'] == $idE){
        $nomE = $elt['nom'];
    }
}

$sql = 'UPDATE `tab_recits_v3` SET `nom_esc` = ?, `titre` = ?, `date_publi` = ?, `lieu_publi` = ?, `mode_publi` = ?, `type_re';
$db = db_connect();
$db->query($sql, [$nomE, $nomR, $dateP, $lieuP, $modeP, $typeR, $com, $idE, $nomS, $lienR, $nomR, $idR]);

$sql = 'DELETE FROM `recit_poly` WHERE `recit_id` = ?';
$db = db_connect();
$db->query($sql, [$idR]);

for($i=0; $i<$nb; $i++){
    $sql = 'INSERT INTO `recit_poly` (`recit_id`, `poly_id`, `type`) VALUES (?, ?, ?)';
    $db = db_connect();
    $db->query($sql, [$idR, $idP[$i], $type[$i]]);
}

return redirect()->to('/recits?search='.$nomR);
}

```

Puis effectuer les traitements dans la base de données. Elle va modifier le récit avec les informations, supprimer les lignes dans `recit_poly` qui sont égales à l'id du récit, puis réinsérer dans la base de données les liaisons entre les polygones et les récits. Elle affichera ensuite la liste des récits.

## Suppression Récit

```

<td>
|   <p><a href="= site_url('Suppr/SupprRecit?esc=' . esc($r['id_auteur']) . '&amp;idR=' . esc($r['id_recit'])) ?&gt;" onclick="return confirm("Supprimer ce récit ?")"&gt;Supprimer&lt;/a&gt;&lt;/p&gt;
&lt;/td&gt;
</pre

```

Lors du clic sur le lien, il va chercher la route `Suppr/SupprRecit` et demander, via une pop-up, une confirmation de la volonté de supprimer le récit.

```
$routes->get('Suppr/SupprRecit', 'Suppr::SupprRecit');
```

La route va appeler la méthode `SupprRecit` du contrôleur `Suppr`.

```

public function SupprRecit()
{
    $db = db_connect();
    $idR = $_GET['idR'];

    $sql = 'DELETE FROM `tab_recits_v3` WHERE `id_recit` = ?';
    $db->query($sql, [$idR]);
    $sql = 'DELETE FROM points WHERE `points`.`id_recit` = ? ';
    $db->query($sql, [$idR]);
    $sql = 'DELETE FROM recit_poly WHERE `recit_id` = ? ';
    $db->query($sql, [$idR]);

    return redirect()->to('/recits');
}

```

La méthode va supprimer tous les points liés au récit ainsi que le récit lui-même et les liaisons entre les récits et les polygones. Elle affichera ensuite la liste des récits.

## Statistique

Lorsque l'on clique sur **Statistiques**, on utilise la route.

```
<?php if ($session->get('is_admin')) : ?>
| <a class="navbar-brand" href="= site_url('statistiques') ?&gt;"&gt;&lt;?= lang('headergeo.nav_bar.statistics') ?&gt;&lt;/a&gt;
&lt;?php endif; ?&gt;</pre
```

Cette route nous renvoie sur la méthode **statistiques** du contrôleur **Admin**.

```
$routes->match(['get', 'post'], 'statistiques', 'Admin::statistiques');
```

Le contrôleur va retourner les vues dans **resclaves/statistique.php** ainsi que le header.

```
public function statistiques()
{
    $session = \Config\Services::session();

    if ($session->has('is_admin') && $session->get('is_admin') === true) {
        return view('resclaves/header')
            . view('resclaves/statistiques');
    } else {
        return redirect()->to('/map');
    }
}
```

## Langue

En haut à droite du site web, il est possible de changer la langue du site. Quand on choisit une langue, le code fait des requêtes pour remplacer tous les champs où l'on retrouve **lang()**.

```
<a href="= site_url() . "map" ?&gt;"&gt;&lt;?php echo lang('headergeo.nav_bar.home')?&gt;&lt;/a&gt;
&lt;hr /&gt;
&lt;a href="<?= site_url() . "recits" ?&gt;"&gt;&lt;?php echo lang('headergeo.nav_bar.list_narratives')?&gt;&lt;/a&gt;
&lt;hr /&gt;</pre
```

Pour cela, il va chercher dans le fichier **headergeo.php** et il va chercher la ligne **nav\_bar.home** ou **nav\_bar.list\_narrative**.

```
<?php

return [
    'nav_bar' => [
        'home' => 'Home',
        'list_narratives' => 'List of narratives',
        'statistics' => "Statistics"
    ],
    'title' => 'Slave narratives',
    'isConnected' => ' (Administrator)',
    'subtitle' => 'Every voice needs to be heard'
];

```

Il existe la même chose pour le français. Cela permet d'avoir une traduction rapide et précise.

## Sidebar

Ouvrir le menu

Rechercher un type de lieu

Selectionner un type de lieu

Rechercher

Rechercher un récit

Rechercher un récit

Rechercher

Pays et villes présents dans le récit

- Lieu de naissance
- Lieu de publication des récits
- Lieu de vie
- Lieu de décès
- Lieu d'esclavage
- Naissance d'esclavage
- Vie et décès
- Esclavage, vie et décès
- Naissance, esclavage, vie et décès
  
- Frontières étatsuniennes

Dans le sidebar, on peut trouver plusieurs fonctionnalités :

## Sélectionner un Type de Lieu

Le premier menu déroulant où l'on peut rechercher par type de lieu permet d'afficher tous les

points du type demandé.

```
<form action=<?= base_url(); ?>/map/places" method="post">
    <?= csrf_field() ?>
    <select name="select_place" id="select">
        <option selected disabled hidden style='display: none' value=''><?= lang('sidebar.location.select_location_type')?></option>

        <?php if($type == 'naissance'){ ?>
            <option value='naissance' selected> <?= lang('sidebar.location.birth')?> </option>
        <?php } else{ ?>
            <option value='naissance'> <?= lang('sidebar.location.birth')?> </option>
        <?php } if($type == 'publication'){?>
            <option value='publication' selected> <?= lang('sidebar.location.publication')?> </option>
        <?php } else{ ?>
            <option value='publication'> <?= lang('sidebar.location.publication')?> </option>

        <?php } if($type == 'deces'){?>
            <option value='deces' selected> <?= lang('sidebar.location.death')?> </option>
        <?php } else{ ?>
            <option value='deces'> <?= lang('sidebar.location.death')?> </option>
        <?php } if($type == 'esclavage'){?>
            <option value='esclavage' selected> <?= lang('sidebar.location.slavery')?> </option>
        <?php } else{ ?>
            <option value='esclavage'> <?= lang("sidebar.location.slavery")?> </option>
        <?php } if($type == 'lieuvie'){?>
            <option value="lieuvie" selected> <?= lang("sidebar.location.location_life")?> </option>
        <?php } else{ ?>
            <option value="lieuvie"> <?= lang("sidebar.location.location_life")?> </option>
        <?php ?>
    </select>

    <br><br>

    <input id="cc" type="submit" value="<?= lang('sidebar.search_button')?>" />
</form>
```

On peut voir le formulaire avec plein de **if**. Cela permet de définir le type de point recherché. Puis la route est définie dans l'action du formulaire et non dans le bouton **submit**, mais le principe reste le même.

```
$routes->match(['get', 'post'], '/map/places', [Map::class, 'index']);
```

La route nous indique que l'on va utiliser la méthode **index** de la classe **Map**, comme si l'on voulait revenir à la carte.

```

// affichage si l'utilisateur choisit le formulaire selon les lieux
else if ($this->request->getPost('select_place')) {

    $data = [
        'type' => $this->request->getPost('select_place')
    ];

    $data = [
        'points' => $model->getPoints(),
        'place' => $model5->search_place($data),
        'aires' => $model4->getAiresAut(),
        'roy_afr' => $model3->getRoyAfr(),
        'type' => $this->request->getPost('select_place')
    ];
}

return view('resclaves/header')
    . view('resclaves/style')
    . view('templates/sidebar', $data)
    . view('resclaves/places', $data)
    . view('templates/footer_resc');
}

```

Sauf que cette fois-ci, on ne va pas utiliser la dernière clause du `if` mais la clause où `select_place` est défini. Notre formulaire va définir `select_place` avec une valeur, ce qui voudra dire que l'on a utilisé le formulaire. Suite à cela, le code va retourner les vues définies avec comme information dans `data`, les points qui sont du type choisi ainsi que les territoires à afficher sur la carte.

## Sélectionner un Récit

Le deuxième menu déroulant est un menu où l'on peut choisir quel récit on veut afficher sur la carte.

```

<form action="= base_url(); ?&gt;/map/recits" method="post"&gt;
    &lt;?= csrf_field() ?&gt;
    &lt;select name="select_recit" id="select"&gt;

        &lt;option&gt;&lt;?= lang('sidebar.narrative.search_narrative') ?&gt;&lt;/option&gt;
        &lt;?php
        $nbt = count($points);
        if (!isset($selec) || $selec === null) {
            $selec = '';
        }

        foreach ($points as $p) {
            if($p['id_recit'] == $selec){?&gt;
                &lt;option value=&lt;?php echo $p['id_recit'] ?&gt; selected&gt;
                    &lt;?php echo $p['nom_esc'], ' (', $p['date_publi'], ')' ?&gt; &lt;/option&gt;

            &lt;?php }else{ ?&gt;
                &lt;option value=&lt;?php echo $p['id_recit'] ?&gt; &gt;
                    &lt;?php echo $p['nom_esc'], ' (', $p['date_publi'], ')' ?&gt; &lt;/option&gt;
            &lt;?php     }
        }?&gt;

    &lt;/select&gt;

    &lt;br&gt;&lt;br&gt;

    &lt;input id="cc" type="submit" value="<?= lang('sidebar.search_button')?&gt;" /&gt;
&lt;/form&gt;
</pre

```

La route nous renvoie sur la méthode `index` du contrôleur `Map`.

```
$routes->match(['get', 'post'], '/map/recits', [Map::class, 'index']);
```

Et cette fois-ci, on va utiliser le premier `if` de la méthode car le formulaire a défini `select_recit`.

```

if ($this->request->getPost('select_recit')) {

    $data2 = [
        'id_recit' => $this->request->getPost('select_recit')
    ];

    $list = $model7->search_recit_poly($data2);

    $data2 = [
        'points' => $model->getPoints(),
        'couche' => $model2->search_adv($data2),

        'aires' => $model4->getAiresAut(),
        'roy_afr' => $model3->getRoyAfr(),
        'pts' => $model5->search_pts($data2),
        'poly' => $model6->search_poly($list),
        'selec' => $this->request->getpost('select_recit')
    ];
}

DatabaseUtils::insertVisit('map_recits');
return view('resclaves/header')
    . view('resclaves/style')
    . view('templates/sidebar', $data2)
    . view('resclaves/accueil2', $data2)
    . view('templates/footer_resc');

}

```

La méthode va retourner les vues demandées avec toutes les informations liées à un récit.

## Menu de Gestion

Dans le menu de gestion, il y a deux groupes de liens :

```

<div id="dropdown" style="display: none;">
    <ul>
        <?php if($session->get('is_admin')): ?>
            <li><a href="/creercompte"><?= lang('sidebar.create_account_button') ?></a></li>
            <li><a href="/ajout_point"><?= lang('sidebar.add_point_button') ?></a></li>
            <li><a href="/ajout_recit"><?= lang('sidebar.add_narrative_button') ?></a></li>
            <li><a href="/ajout_poly"><?= lang('sidebar.add_polygon_button') ?></a></li>
            <li><a href="/ajout_esclave"><?= lang('sidebar.add_slave_button') ?></a></li>
            <li><a href="/choix_esclave"><?= lang('sidebar.modify_slave_button') ?></a></li>
            <li><a href="/suppr_esclave"><?= lang('sidebar.delete_slave_button') ?></a></li>
        <?php endif ?>

        <li><a href="= $session-&gt;get('is_admin') ? '/deconnexion' : '/connexion' ?&gt;"&gt;
            | onclick="<?= $session-&gt;get('is_admin') ? 'return confirmLogout()' : '' ?&gt;"&gt;
            &lt;?= $session-&gt;get('is_admin') ? lang('sidebar.logout_button') : lang('sidebar.login_button') ?&gt;
        &lt;/a&gt;&lt;/li&gt;
    &lt;/ul&gt;
&lt;/div&gt;
</pre

```

Le premier groupe qui s'affiche seulement si l'utilisateur est connecté et un deuxième où il affiche soit **déconnexion** quand on est connecté ou **connexion** quand on ne l'est pas.

## Déconnecté

Commençons par le début, et donc quand on arrive sur le site web, l'utilisateur est déconnecté.

### Connexion

Pour se connecter, il faut cliquer sur le bouton suivant :

```

<li><a href="= $session-&gt;get('is_admin') ? '/deconnexion' : '/connexion' ?&gt;"&gt;
    | onclick="<?= $session-&gt;get('is_admin') ? 'return confirmLogout()' : '' ?&gt;"&gt;
    &lt;?= $session-&gt;get('is_admin') ? lang('sidebar.logout_button') : lang('sidebar.login_button') ?&gt;
&lt;/a&gt;&lt;/li&gt;
</pre

```

Comme l'utilisateur n'est pas connecté, il utilisera la route **/connexion** :

```
$routes->match(['get', 'post'], '/connexion', [Admin::class, 'showconnexion']);
```

Cette route renvoie vers la méthode **showconnexion** du contrôleur **Admin** :

```

public function showconnexion()
{
    DatabaseUtils::insertVisit('connexion');

    return view('resclaves/header')
        . view('resclaves/connexion');
}

```

La méthode renvoie la vue de connexion (**connexion.php**).

Sur la page de connexion, on peut remplir deux champs du formulaire (**username**, **password**) :

```
<div class="login-container">
    <h2><?= lang('connexion.title') ?></h2>
    <form action=<?= site_url('/Admin/login') ?>" method="post">
        <div class="input-group">
            <label for="username"><?= lang('connexion.username')?></label>
            <input type="text" id="username" name="username" required>
        </div>
        <div class="input-group">
            <label for="password"><?= lang('connexion.password')?></label>
            <input type="password" id="password" name="password" required>
        </div><br>
        <button type="submit"><?= lang('connexion.login_button')?></button>
    </form>
</div>
```

Le formulaire enverra les données en utilisant sa route `/Admin/login` :

```
$routes->post('Admin/login', 'Admin::login');
```

Cette route mènera à la méthode `login` du contrôleur `Admin` :

```

public function login()
{
    // Obtenez les données de formulaire
    $username = $this->request->getPost('username');
    $password = $this->request->getPost('password');

    $hashpassword = hash('sha256', $password, true);
    $hashhexa = bin2hex($hashpassword);

    $model = model(ModelAdmin::class);

    $storedPassword = $model->getPass($username);

    if ($hashhexa == $storedPassword & $storedPassword !== null) {
        // Démarrer la session
        $session = \Config\Services::session();
        // Définir is_admin à true
        $session->set('is_admin', true);

        // Rediriger vers la page d'administration
        return redirect()->to('/map');
    } else {
        // Gérer l'échec de la connexion
        // ...
        // afficher un message d'erreur

        //commenté pour afficher les mdp
        return redirect()->to('/connexion');
    }
}

```

La méthode récupérera les champs du formulaire, hashera le mot de passe, et le comparera au mot de passe reçu dans la requête en utilisant

## Ajout Point

## Ajout Récit

## Ajout Polygone

## Ajout Esclave/Auteur

## Modification d'un Esclave/auteur

## Suppression d'un Esclave/auteur

# Footer



Dans le footer, on retrouve deux fonctionnalités : le contact avec la possibilité d'envoyer un mail à l'adresse mail du site et une page avec des informations et remerciements.

```
<br>
<footer>
| <div class="text-center p-3">
| | <a id="lienfoot" href="<% site_url()."about" %>"><% lang('footer_resc.about') %></a>
| | <a id="lienfoot" href="<% site_url()."contact" %>"><% lang('footer_resc.contact') %></a>
| </div>
</footer>

<style>
| footer{
| | background-color:#a0512db9;
| }
</style>
</body>
</html>
```

# Contact

Dans la route, on appelle la méthode `contact` du contrôleur `Map`.

```
$routes->match(['get', 'post'],'/contact', [Map::class, 'contact']);
```

Cette méthode retourne la page `contact`.

```
public function contact()
{
    DatabaseUtils::insertVisit('contact');

    return view('resclaves/header')
        . view('resclaves/contact_resc')
        . view('templates/footer_resc');
```

Dans cette page, on retrouve un formulaire où l'on peut remplir les informations à transmettre dans le mail.

```

<form id="myForm">
  <div class="login-container">
    <h2><?= lang('contact_resc.title') ?></h2>
    <div class="input-group">
      <label for="name"><?= lang('contact_resc.name') ?></label>
      <input type="text" id="name" name="name" required>
    </div>
    <div class="input-group">
      <label for="email"><?= lang('contact_resc.email') ?></label>
      <input type="email" id="email" name="email" required>
    </div>
    <div class="input-group">
      <label for="message"><?= lang('contact_resc.message') ?></label>
      <textarea id="message" name="message" rows="5" style="resize: none;" required></textarea>
    </div>
    <button type="button" onclick="sendMail()"><?= lang('contact_resc.send_button') ?></button>
  </div>
</form>

```

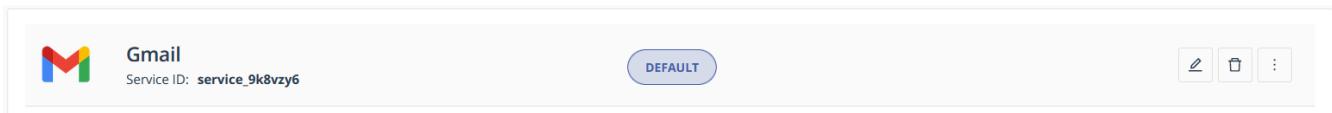
Mais on trouve aussi du JavaScript pour faire l'envoi du mail.

```

<script type="text/javascript">
  (function() {
    emailjs.init("RVs1DfIzp08lrBGl"); // Utilisez le Service ID "service_9k8vzy6"
  })();
</script>

```

Le premier bloc définit le service à utiliser par son identifiant.



Et le deuxième bloc contient l'envoi du mail avec le template à utiliser. Le template permet de pré-structurer le mail avec les informations fournies dans le mail.

```

<script>
    function sendMail() {
        // Récupérer la valeur de l'e-mail
        var email = document.getElementById("email").value;

        // Vérifier si l'e-mail est valide en utilisant une regex
        var emailRegex = /^[a-zA-Z0-9!#$%&'*+\/=?^_`{|}~-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9]{2,4}$/;
        if (!emailRegex.test(email)) {
            alert("Cet email n'est pas valide");
            return; // Arrêter l'envoi du formulaire si l'e-mail n'est pas valide
        }

        // Si l'e-mail est valide, continuer avec l'envoi du formulaire
        var params = {
            from_name: document.getElementById("name").value,
            email_id: email,
            message: document.getElementById("message").value
        };

        emailjs.send("service_9k8vzy6", "template_5e82ku5", {
            from_name: document.getElementById("name").value,
            email_id: document.getElementById("email").value,
            message: document.getElementById("message").value
        }).then(function (res) {
            alert("Message envoyé !");
        }).catch(function (error) {
            console.error("Erreur lors de l'envoi de l'e-mail : ", error);
        });
    }
</script>

```



## Information

Pour les informations du site web :

```
$routes->match(['get', 'post'],'/about', [Map::class, 'about']);
```

Le lien renvoie vers la méthode `about` du contrôleur `Map`.

```

public function about()
{
    DatabaseUtils::insertVisit('about');

    return view('resclaves/header')
        .view('resclaves/about_resc')
        .view('templates/footer_resc');
}

```

La page contient juste des informations et des remerciements.