

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	11
Autoch2	11
Fullpt_1	11
Link	11
Map3	11
Page	12
Points	12
Polygone	12
Pts_publication	13
Recit_poly	13
Roy_afr	13
Tab_auteurs	14
Tab_recits_v3	14
User	15
Visite	15
Accueil	15
Map	15
Header	17
Accueil	18
Liste Récit	20
Récit	23
Modification Récit	23
Suppression Récit	26
Récit en Détail	27
Affichage d'un récit	29
Ajout d'un lien	29
Api Zotero	32
Statistique	36

Langage	36
Sidebar	37
Sélectionner un type de lieu	38
Sélectionner un récit	40
Menu de gestion	42
Déconnecté	43
Connexion	43
Ajout Point	45
Ajout Récit	46
Ajout Polygone	50
Ajout Esclave/Auteur	53
Modification d'un Esclave/Auteur	56
Suppression d'un Esclave/Auteur	60
Footer	61
Contacts	62
Information	64

Architecture Code Igniter 4

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

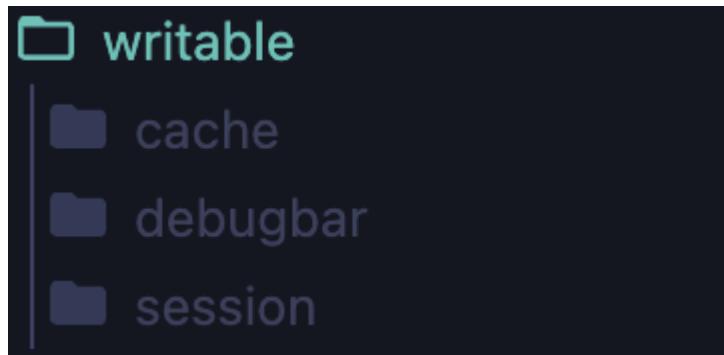
```
> app
> public
> system
> writable
⚙ .env
↳ 0index.html
🐘 0index.php
composer.json
≡ env
≡ htaccess
LICENSE
package.json
≡ phpunit.xml.dist
🐘 preload.php
README.md
≡ spark
```

Dans le projet, on retrouve 5 parties :

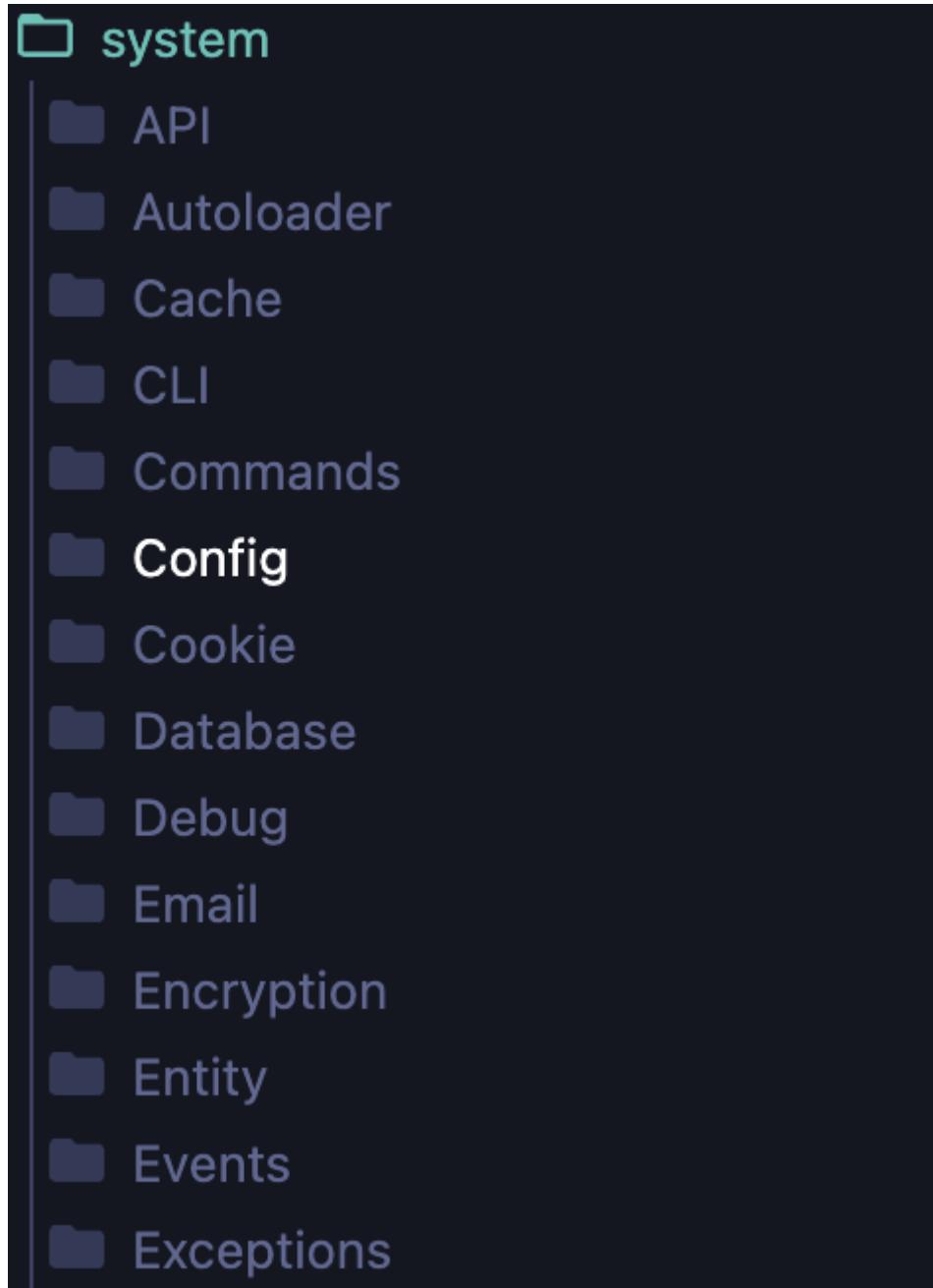
- La première est la configuration du projet Code Igniter avec le fichier .env à modifier pour le fonctionnement de votre site.



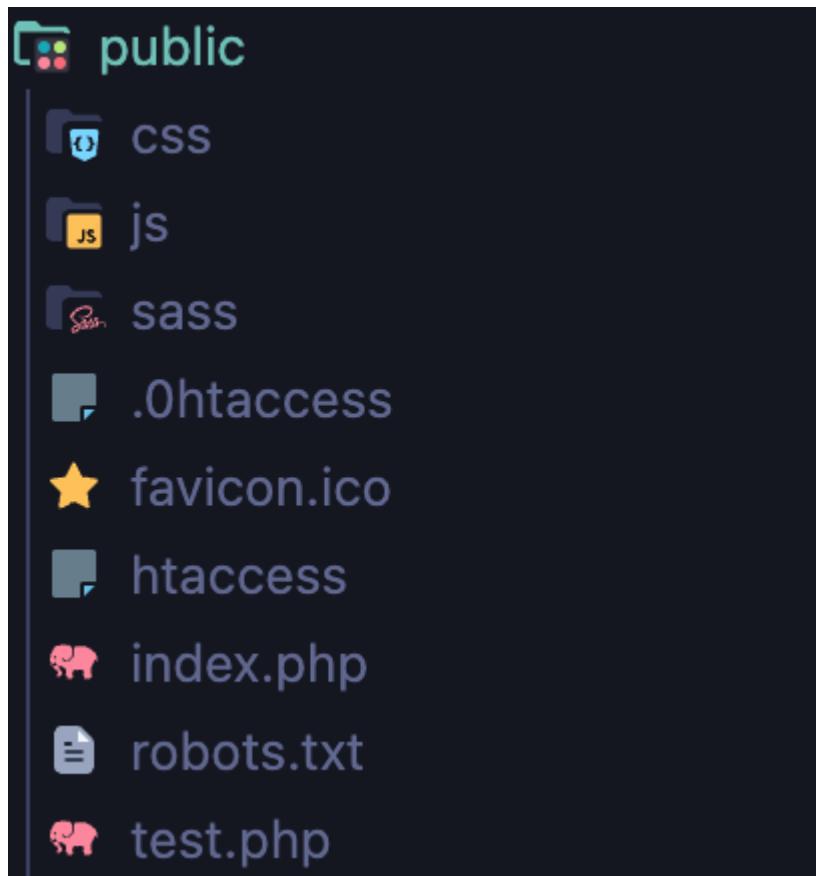
- La deuxième est les writables, ils stockent toutes les informations liées à des problèmes.



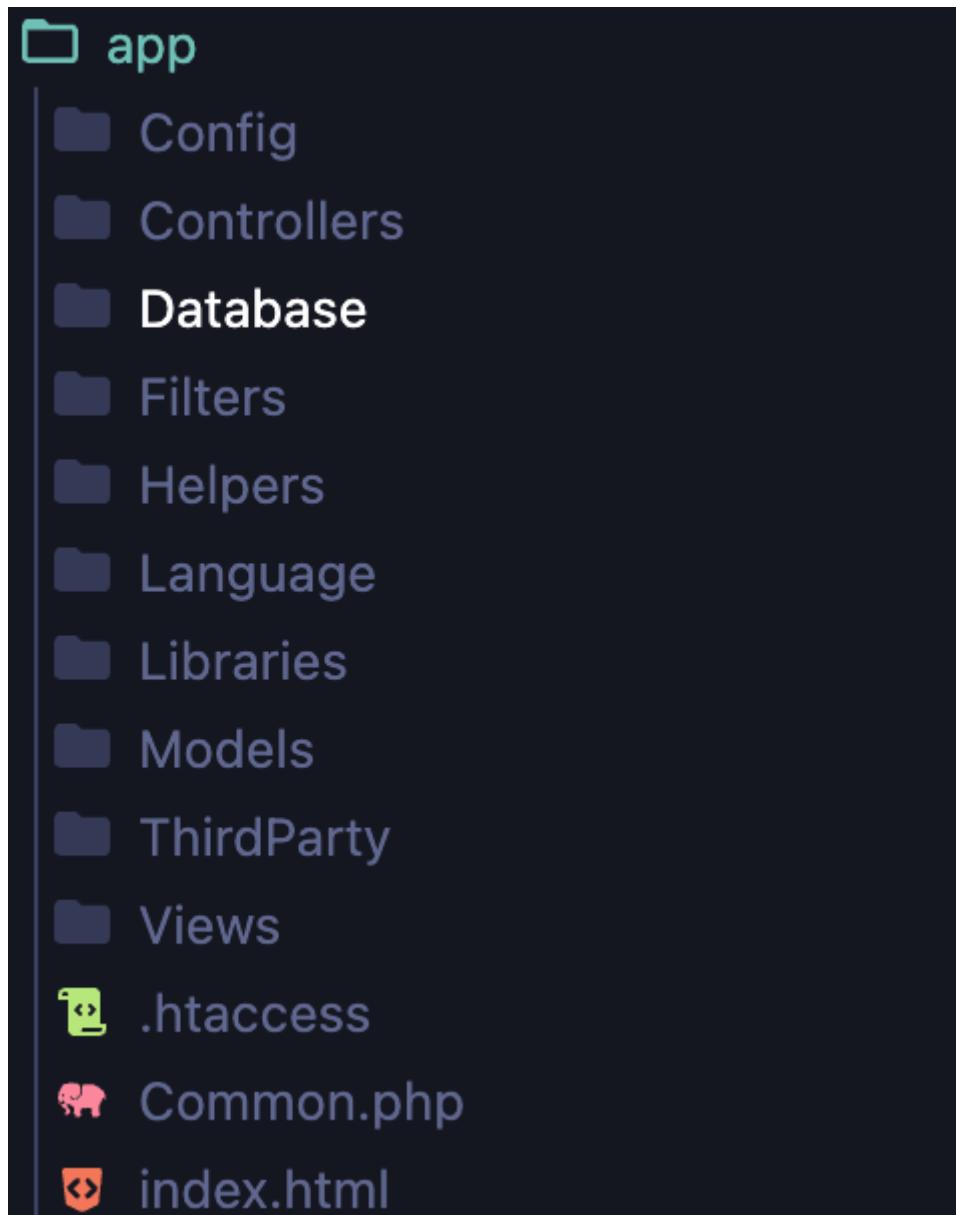
- Le troisième est le système. Tout le fonctionnement du site est contenu dedans.



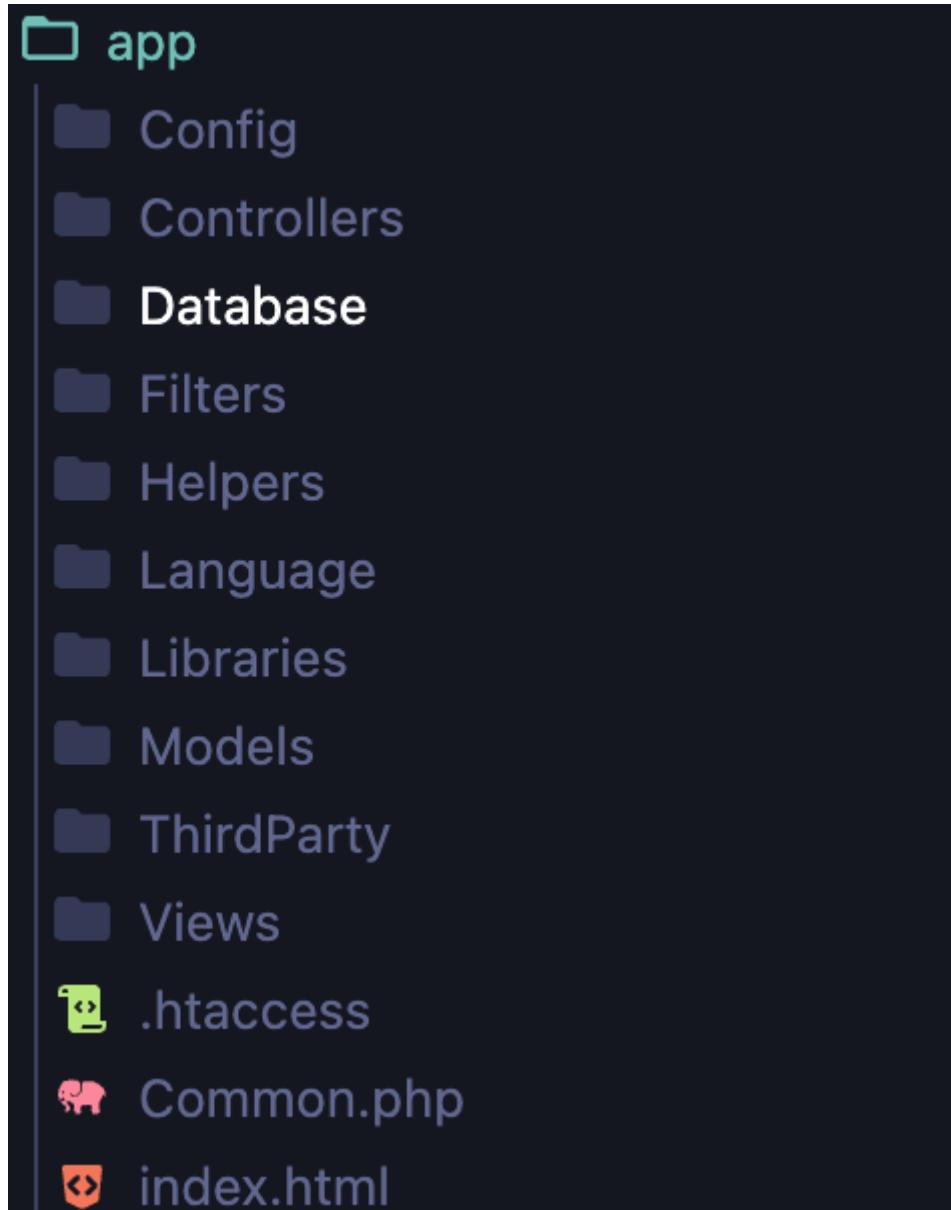
- Le quatrième est le dossier public. Dans celui-ci, on peut retrouver les fichiers css, js ou tout autre langage utilisé.



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



Rentrons 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

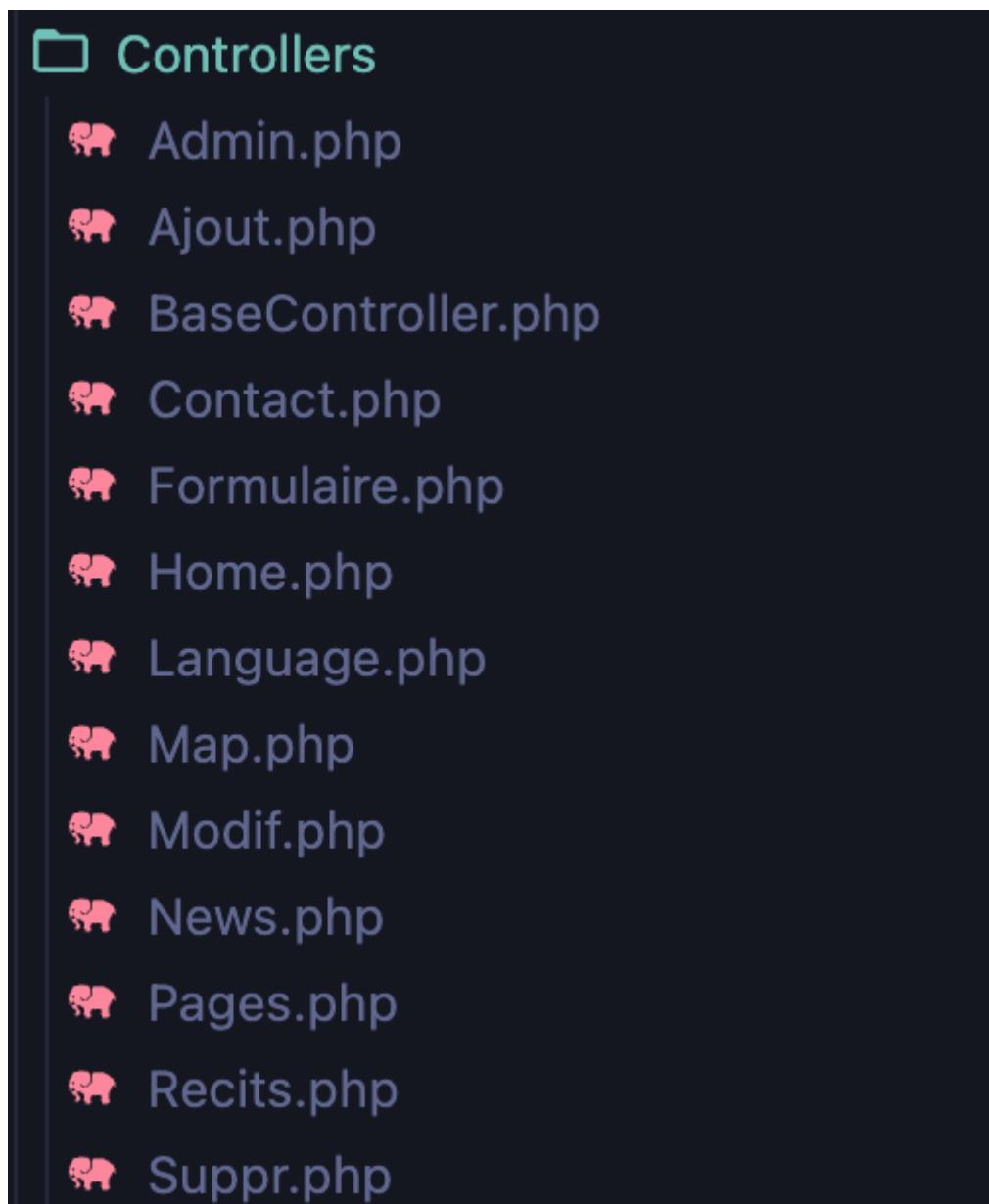
Generators.php

Honeypot.php

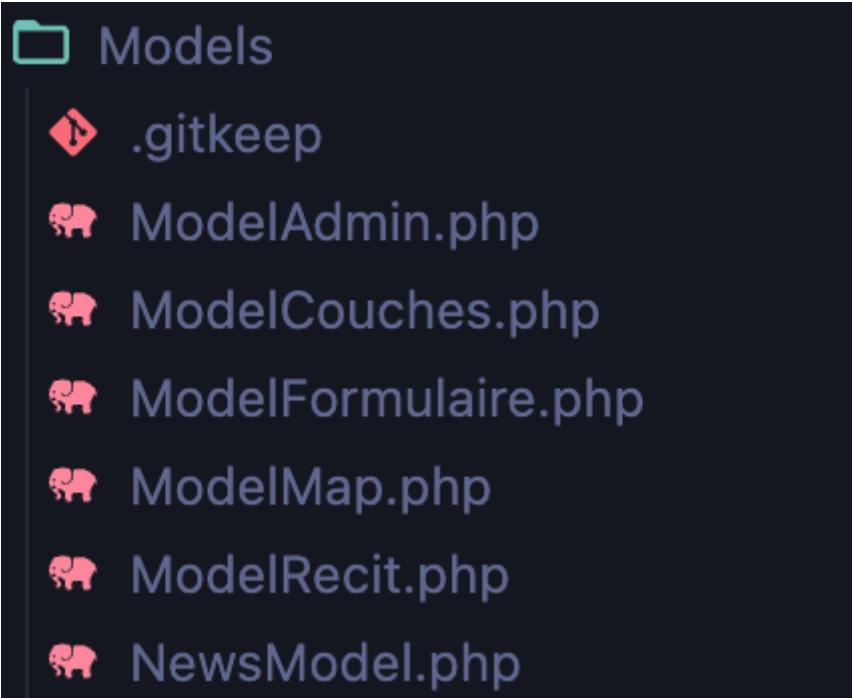
Images.php

Kint.php

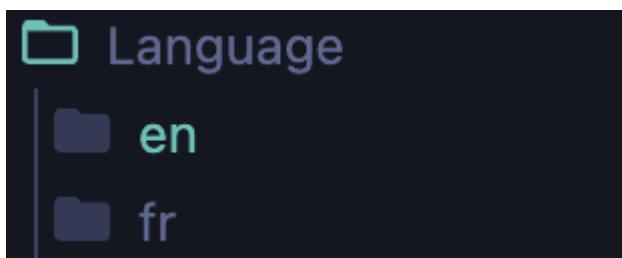
- Le dossier contrôleur, où on trouve les contrôleurs qui permettent la gestion des vues. Les contrôleurs importent les méthodes qui sont définies dans les modèles.



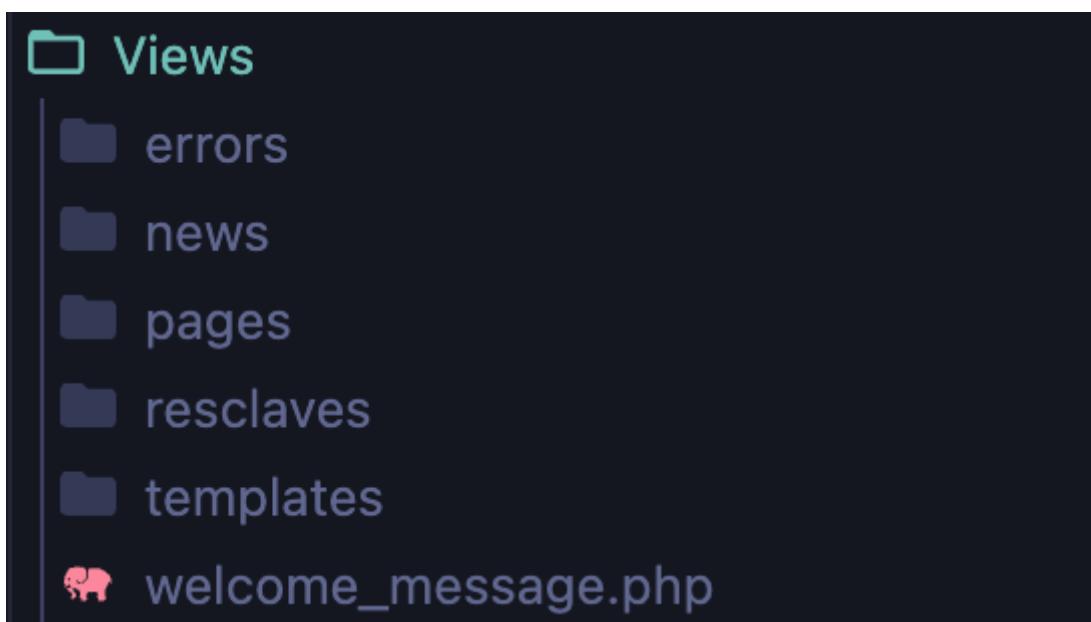
- Le dossier modèle, où il y a tous les modèles de méthodes. On y retrouve des méthodes de requête en base de données ou de récupération de données qui seront appliquées dans les contrôleurs.



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



- 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.



Base de données

Voici la documentation technique pour la base de données liée au site web.

Autoch2

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	WKT	varchar(2622)	utf8mb4_general_ci		Oui	NULL		
2	id 	int(1)			Non	Aucun(e)		
3	id_style 	int(1)			Oui	NULL		
4	geoj	text	utf8mb4_general_ci		Oui	NULL		

Cette table contient des polygones en Amérique.

Fullpt_1

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	WKT	varchar(43)	utf8mb4_general_ci		Oui	NULL		
2	ville	varchar(22)	utf8mb4_general_ci		Oui	NULL		
3	layer	varchar(27)	utf8mb4_general_ci		Oui	NULL		
4	id 	int(11)			Non	Aucun(e)		AUTO_INCREMENT
5	id_recit 	varchar(5)	utf8mb4_general_ci		Oui	NULL		
6	type	varchar(100)	utf8mb4_general_ci		Oui	NULL		
7	geoj	text	utf8mb4_general_ci		Oui	NULL		

Cette table contient des points.

Link

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	reference 	varchar(70)	utf8mb4_general_ci		Non	Aucun(e)		
2	link 	varchar(110)	utf8mb4_general_ci		Non	Aucun(e)		

Cette table contient des liens pour les récits.

Map3

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	id	varchar(9)	utf8mb4_general_ci		Oui	NULL		
2	label	varchar(62)	utf8mb4_general_ci		Oui	NULL		
3	category	varchar(14)	utf8mb4_general_ci		Oui	NULL		
4	state	varchar(5)	utf8mb4_general_ci		Oui	NULL		
5	layer	varchar(10)	utf8mb4_general_ci		Oui	NULL		
6	id_recit	varchar(8)	utf8mb4_general_ci		Oui	NULL		
7	id_1	varchar(4)	utf8mb4_general_ci		Non	Aucun(e)		
8	geoj	varchar(9886)	utf8mb4_general_ci		Oui	NULL		
9	country	varchar(20)	utf8mb4_general_ci		Oui	NULL		

Cette table contient des polygones.

Page

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	id	int(11)			Non	Aucun(e)		AUTO_INCREMENT
2	nom	varchar(255)	utf8mb4_general_ci		Non	Aucun(e)		
3	description	varchar(30)	utf8mb4_general_ci		Non	Aucun(e)		

Cette table liste toutes les pages du site web.

Points

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	WKT	varchar(43)	utf8mb4_general_ci		Oui	NULL		
2	ville	varchar(22)	utf8mb4_general_ci		Oui	NULL		
3	layer	varchar(27)	utf8mb4_general_ci		Oui	NULL		
4	id	int(11)			Non	0		
5	id_recit	varchar(5)	utf8mb4_general_ci		Non	Aucun(e)		
6	type	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)		
7	geoj	text	utf8mb4_general_ci		Oui	NULL		

Cette table contient les points liés aux récits.

Polygone

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	name_1	varchar(70)	utf8mb4_general_ci		Non	Aucun(e)		
2	name	varchar(110)	utf8mb4_general_ci		Non	Aucun(e)		
3	label	varchar(80)	utf8mb4_general_ci		Non	Aucun(e)		
4	category	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)		
5	state	varchar(50)	utf8mb4_general_ci		Non	Aucun(e)		
6	geoj	mediumtext	utf8mb4_general_ci		Non	Aucun(e)		
7	id 	int(11)			Non	Aucun(e)		AUTO_INCREMENT

Cette table contient les polygones liés aux récits.

Pts_publication

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	WKT	varchar(42)	utf8mb4_general_ci		Oui	NULL		
2	id  	int(2)			Non	Aucun(e)		
3	ville 	varchar(18)	utf8mb4_general_ci		Oui	NULL		
4	layer 	varchar(52)	utf8mb4_general_ci		Oui	NULL		
5	id_recit 	varchar(52)	utf8mb4_general_ci		Oui	NULL		
6	geoj	text	utf8mb4_general_ci		Oui	NULL		
7	fid	int(11)			Oui	NULL		

Cette table contient les points de publication.

Recit_poly

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	recit_id 	varchar(3)	utf8mb4_general_ci		Non	Aucun(e)		
2	poly_id 	int(3)			Non	Aucun(e)		
3	type 	varchar(10)	utf8mb4_general_ci		Non	Aucun(e)		

Cette table contient les liens entre les polygones et les récits.

Roy_afr

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	wkt	text	utf8mb4_general_ci		Oui	NULL		
2	id  	varchar(70)	utf8mb4_general_ci		Non	Aucun(e)		
3	noms  	varchar(100)	utf8mb4_general_ci		Non	Aucun(e)		
4	geoj	text	utf8mb4_general_ci		Oui	NULL		

Cette table contient les polygones des royaumes africains.

Tab_auteurs

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	nom 	varchar(58)	utf8mb4_general_ci		Oui	NULL		
2	naissance	varchar(27)	utf8mb4_general_ci		Oui	NULL		
3	lieu_naissance 	varchar(38)	utf8mb4_general_ci		Oui	NULL		
4	deces	varchar(10)	utf8mb4_general_ci		Oui	NULL		
5	lieu_deces 	varchar(29)	utf8mb4_general_ci		Oui	NULL		
6	lieu_esclavage 	varchar(53)	utf8mb4_general_ci		Oui	NULL		
7	moyen_lib	varchar(117)	utf8mb4_general_ci		Oui	NULL		
8	lieuvie_ap_lib 	varchar(72)	utf8mb4_general_ci		Oui	NULL		
9	origine_parents	varchar(164)	utf8mb4_general_ci		Oui	NULL		
10	militant_abolitionniste	varchar(37)	utf8mb4_general_ci		Oui	NULL		
11	particularites	varchar(113)	utf8mb4_general_ci		Oui	NULL		
12	plrs_recits	varchar(3)	utf8mb4_general_ci		Oui	NULL		
13	id_auteur  	varchar(3)	utf8mb4_general_ci		Non	Aucun(e)		
14	op_source	varchar(164)	utf8mb4_general_ci		Non	Aucun(e)		

Cette table contient les auteurs du site web.

Tab_recits_v3

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	nom_esc	varchar(47)	utf8mb4_general_ci		Oui	NULL		
2	titre	varchar(431)	utf8mb4_general_ci		Oui	NULL		
3	date_publi	int(4)			Oui	NULL		
4	lieu_publi	varchar(25)	utf8mb4_general_ci		Oui	NULL		
5	mode_publi	varchar(75)	utf8mb4_general_ci		Oui	NULL		
6	type_recit	varchar(5)	utf8mb4_general_ci		Oui	NULL		
7	historiographie	varchar(833)	utf8mb4_general_ci		Oui	NULL		
8	id_auteur	varchar(3)	utf8mb4_general_ci		Oui	NULL		
9	id_recit 	varchar(3)	utf8mb4_general_ci		Non	Aucun(e)		
10	scribe_editeur	varchar(125)	utf8mb4_general_ci		Oui	NULL		
11	lien_recit	varchar(224)	utf8mb4_general_ci		Oui	NULL		
12	debut_titre	varchar(431)	utf8mb4_general_ci		Oui	NULL		

Cette table contient les récits du site.

User

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	user 	varchar(32)	utf8mb4_general_ci		Non	Aucun(e)		
2	pwd	varchar(128)	utf8mb4_general_ci		Non	Aucun(e)		

Cette table contient les utilisateurs pour se connecter.

Visite

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
1	id 	int(11)			Non	Aucun(e)		AUTO_INCREMENT
2	jour	timestamp			Non	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()
3	id_page 	int(11)			Oui	NULL		

Cette table liste les connexions aux pages pour les statistiques de visite.

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

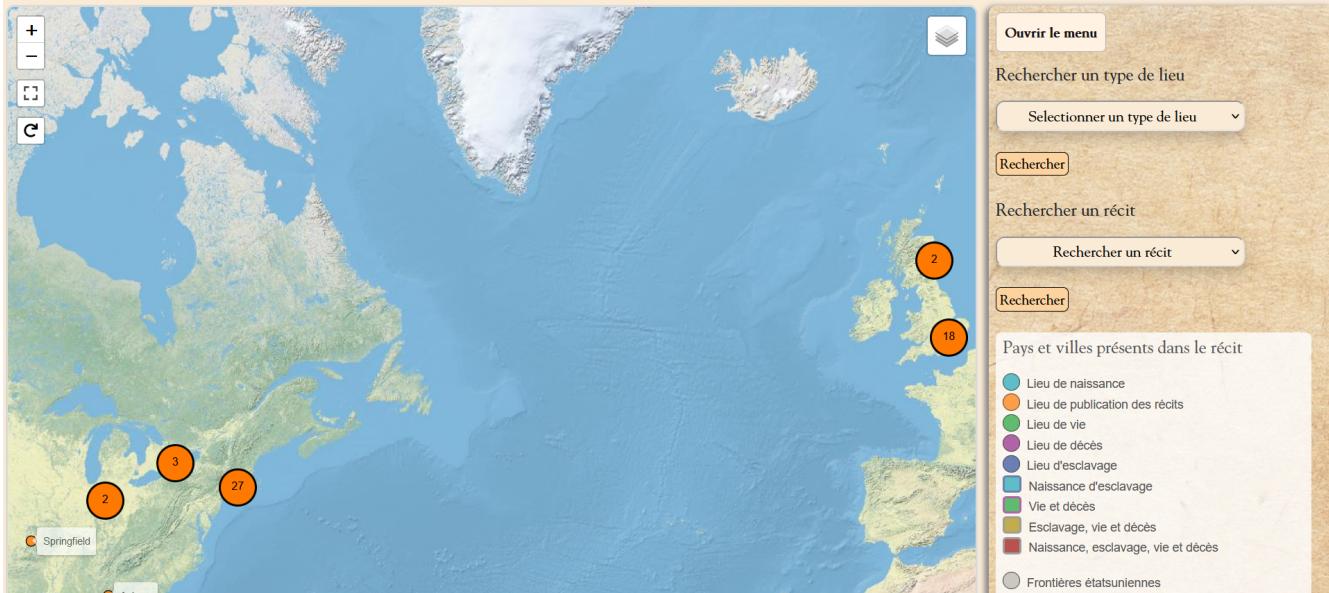
Accueil

Map

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

Récits d'esclaves (Administrateur)

Chaque voix doit être entendue



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://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
  maxZoom: 17,
  attribution: 'Map data: © openStreetMap contributors'
});

var OpenStreetMap_Mapnik = L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
  maxZoom: 19,
  attribution: '&copy; © OpenStreetMap 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" ?>><?= lang('headergeo.nav_bar.home')?></a>
  <a class="navbar-brand" href=<?= site_url() . "recits" ?>><?= lang('headergeo.nav_bar.list_narratives')?></a>

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

  <div class="language">
    <a href="#" id="changeLanguageEN" class="language-link<?php echo ($session->get('locale') === 'en') ? 'language-link-active' : ''?>"><?= lang('headergeo.nav_bar.en')?></a>
    <a href="#" id="changeLanguageFR" class="language-link<?php echo ($session->get('locale') === 'fr') ? 'language-link-active' : ''?>"><?= lang('headergeo.nav_bar.fr')?></a>
  </div>
</nav>
```

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 propre valeur
  form.appendChild(idInput);

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

  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;&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;&lt;?= 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="re
            &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=' . esc($r['id_auteur']) . '&amp;idR=' . esc($r['id_recit'])) ?&gt;"&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 cherché la route "Suppr/SupprRecit" et demandé avec 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(Suppr.php)

```

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 tout les points liées au récit ainsi que le récit et les liaisons être les récits et les polygones. Puis va afficher la liste des récits.

Récit en Détail

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

Lorsqu'on clique sur un récit dans la liste, le formulaire appelle la route recits/ + l'id du récit.

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

La route appelle la méthode view du contrôleur Recits.

```
public function view($idrec = null){

    $model = model(ModelRecit::class);
    $texteArray = $model->getIdRec($idrec);
    $textehisto = $texteArray["historiographie"];

    // Tableau pour stocker les valeurs extraites
    $apiValues = array();

    // Extrait les informations entre parenthèses
    $pattern = '/\(([^\)]+)\)\/';
    preg_match_all($pattern, $textehisto, $matches);
```

La méthode récupère les informations du récit sélectionné dans les paramètres puis fait une recherche de toutes les informations liées au récit en paramètre. Puis le champ "historiographie" est récupéré à part pour être modifié.

Et enfin, on définit un pattern () et on récupère toutes les occurrences où il y a des parenthèses.

```
// Parcourt les correspondances et les traite
foreach ($matches[1] as $match) {
    // Divise le texte en segments en utilisant la virgule
    $segments = explode(',', $match);

    // Vérifie qu'il y a au moins trois segments (auteur, titre, année)
    if (count($segments) >= 3) {
        $auteur = trim($segments[0]);
        $titre = trim($segments[1]);
        $annee = (int) trim($segments[2]);

        // Génère le lien avec l'appel JavaScript
        $lien = "<a href='javascript:void(0);' onclick=\"$afficherPopup('$titre')\">($auteur, $titre, $annee)</a>";

        // Remplace le texte entre parenthèses par le lien
        $textehisto = str_replace("($match)", $lien, $textehisto);
    }
}
```

Pour chaque occurrence des parenthèses, on va diviser le texte avec comme séparateur la virgule, et l'affecter à \$segments.

Si le segment a trois éléments, cela veut dire que dans la parenthèse on retrouve (Auteur, Titre raccourci, page). Puis on veut récupérer chaque élément et on va générer un string qui appelle la fonction javascript permettant de récupérer les infos de Zotero.

```
if(count($segments) == 2){
    $titre = trim($segments[0]);
    $annee = (int) trim($segments[1]);

    // Génère le lien avec l'appel JavaScript
    $lien = "<a href='javascript:void(0);' onclick=\"afficherPopup('$titre')\">($titre, $annee)</a>";

    // Remplace le texte entre parenthèses par le lien
    $textehisto = str_replace("($match)", $lien, $textehisto);
}
```

Si le nombre d'éléments est égal à 2, cela veut dire qu'il y a le (Titre, page). Mais le principe reste le même.

```
if(count($segments) == 1){
    $titre = trim($segments[0]);
    $sql = 'SELECT * FROM `link` WHERE reference = ?';
    $db = db_connect();
    $result = $db->query($sql, [$titre]);

    if ($result) {
        $row = $result->getRow(); // Obtenez la première ligne de résultat
        if ($row) {
            $reference = $row->reference;
            $link = $row->link;
            $lien = "<a href='$link'>($reference)</a>";
        } else {
            // Aucun résultat trouvé pour le titre donné
            $lien = "<a >($titre)</a>";
        }
    } else {
        // Gérez les erreurs de requête ici, par exemple :
        die("Erreur de requête SQL : " . $db->error());
    }
    $textehisto = str_replace("($match)", $lien, $textehisto);
}
```

Si le nombre d'éléments dans la parenthèse est égal à 1, cela veut dire que c'est un lien et donc on va récupérer les liens stockés dans la table Link. Si le lien existe, il va modifier les pour mettre le lien. Sinon, il ne change rien.

```

// Assigner le tableau $apiValues à $data['api']
$data['api'] = $apiValues;

$data['rec'] = $texteArray;
$data['histo'] = $textehisto;

if (empty($data['rec'])) {
    throw new PageNotFoundException('Cannot find the news item: ' . $idre);
}

DatabaseUtils::insertVisit('fiche_recit');
return view('resclaves/header')
    . view('resclaves/view', $data)
    . view('templates/footer_resc');

```

Puis il affiche la page view avec les informations du récit.

Affichage d'un récit

Dans la page on retrouve toutes les informations liées au récit sélectionné.

Ajout d'un lien

Mais on trouve aussi la possibilité de modifier et supprimer le récit depuis cette page. Mais le principal est une possibilité d'ajouter des liens dans la BD.

```

<?php if ($session->get('is_admin')) : ?>
    <td>
        <p><a href="= site_url('/modif_recit?esc=' . esc($rec['id_auteur']) . '&amp;idR=' . esc($rec['id_recit'])) ?&gt;"&gt;&lt;?= lang('recits') ?&gt;</a&gt;
    &lt;/td&gt;
    &lt;td&gt;
        &lt;p&gt;&lt;a href="<?= site_url('Suppr/SupprRecit?esc=' . esc($rec['id_auteur']) . '&amp;idR=' . esc($rec['id_recit'])) ?&gt;" onclick="retour();&gt;&lt;?= lang('suppr') ?&gt;</a&gt;
    &lt;/td&gt;
    &lt;td&gt;
        &lt;p&gt;&lt;a href="/ajout_link"&gt;Ajouter un lien&lt;/a&gt;&lt;/p&gt;
    &lt;/td&gt;
&lt;?php endif; ?&gt;
</pre

```

Lors du clic sur le lien, la route appelle "/ajout link".

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

La route appelle la méthode ajout_link du contrôleur Ajout.

```

public function ajout_link(){
    return view('resclaves/header')
        .view('resclaves/ajout_link');
}

```

La méthode s'occupe juste d'afficher la page d'ajout.

```
<form id="form1" class="form" action="#" method="post">
    <h2><?= lang('ajout_link.tab1') ?></h2>
    <label for="champ1"><?= lang('ajout_link.tab1_ch1') ?></label>
    <input type="text" id="champ1" name="champ1"><br>
    <label for="champ2"><?= lang('ajout_link.tab1_ch2') ?></label>
    <input type="text" id="champ2" name="champ2"><br><br>
    <a class="retour" href="= site_url('/recits') ?&gt;"&gt;&lt;?= lang('recits.bouton_retour') ?&gt;&lt;/a&gt;&lt;/p&gt;
    &lt;button type="button" id="addLinkButton"&gt;&lt;?= lang('ajout_link.tab1_add') ?&gt;&lt;/button&gt;
&lt;/form&gt;</pre
```

Sur cette page on retrouve deux formulaires. Le premier est juste un formulaire avec 2 champs, un pour le nom du lien et un autre pour le lien.

```
<script>
    document.addEventListener("DOMContentLoaded", function() {
        var refs = [];
        var links = [];

        var addLinkButton = document.getElementById("addLinkButton");
        var linkTable = document.getElementById("linkTable");
        var refInput = document.getElementById("refinput");
        var linkInput = document.getElementById("linkinput");

        addLinkButton.addEventListener("click", function() {
            var champ1Value = document.getElementById("champ1").value;
            var champ2Value = document.getElementById("champ2").value;

            // Créez un tableau pour stocker les données
            var data = [champ1Value, champ2Value];
            // Ajoutez les données au tableau
            refs.push(data[0]);
            links.push(data[1]);

            var newRow = linkTable.insertRow();
            var cell1 = newRow.insertCell(0);
            var cell2 = newRow.insertCell(1);
            var deleteCell = newRow.insertCell(2);

            cell1.innerHTML = data[0];
            cell2.innerHTML = data[1];

            // Ajouter un bouton de suppression pour la nouvelle ligne
            var deleteButton = document.createElement("button");
            deleteButton.textContent = "Supprimer";
            deleteButton.addEventListener("click", function() {
                var index = Array.from(linkTable.rows).indexOf(newRow);
                if (index !== -1) {
                    // Supprimez la ligne du tableau
                    linkTable.deleteRow(index);
                }
            });
        });
    });
</script>
```

Quand on clique sur ajouter, le script javascript s'occupe d'ajouter les informations du 1er

formulaire dans le tableau du deuxième. Cela permet d'ajouter plusieurs liens en même temps.

```
<form id="form2" class="form" action="= site_url('Ajout/InsertLink') ?" method="post">
    <h2><?= lang('ajout_link.tab2') ?></h2>
    <table>
        <thead>
            <tr>
                <th><?= lang('ajout_link.tab2_name') ?></th>
                <th><?= lang('ajout_link.tab2_link') ?></th>
                <th style="position: relative; width: 30%;"><?= lang('ajout_poly.suppr') ?></th>
            </tr>
        </thead>
        <tbody id="linkTable">
            <!-- Les coordonnées seront ajoutées ici -->
        </tbody>
    </table>
    <input type="hidden" name="ref" id="refinput">
    <input type="hidden" name="link" id="linkinput"><br>
    <button type="submit"><?= lang('ajout_link.tab2_send') ?></button>
</form>
```

Le 2ème formulaire sert à stocker et afficher les liens à ajouter. Quand on clique sur validé du formulaire, il appelle la route "Ajout/InsertLink".

```
$routes->post('Ajout/InsertLink', 'Ajout::InsertLink');
```

La route renvoie vers la méthode InsertLink du contrôleur Ajout.

```
public function InsertLink() {
    // Récupérer les données du formulaire
    $data = $this->request->getPost();

    // Décodez les données JSON depuis les chaînes
    $references = json_decode($data['ref']);
    $liens = json_decode($data['link']);
    //var_dump($references);

    // Assurez-vous que les décodages ont réussi
    if ($references !== null && $liens !== null) {
        // Faites le traitement souhaité avec les données
        for ($i = 0; $i < count($references); $i++) {
            $sql = 'INSERT INTO `link` (`reference`, `link`) VALUES (?, ?)';
            $db = db_connect();
            $db->query($sql, [$references[$i], $liens[$i]]);

            return redirect()->to('/recits');
        }
    } else {
        // Gérez l'erreur de décodage JSON
        echo "Erreur lors du décodage JSON des données.";
        return redirect()->to('/recits');
    }
}
```

La méthode va insérer dans la table link les liens créés.

Api Zotero

```
<div id="comm">
    <p style="text-align:center;">
        Commentaires / Historiographie: <br><br>
        <?= html_entity_decode($histo) ?>

</p>
</div>
```

L'api Zotero sera utilisée que dans la partie commentaire.

```
<script>
function afficherPopup(choix) {
    var apiKey = "E7a5WJBnmii1HdXPtMVRZcG1";
    var userId = "5400206";
    var Apidata = [];
    var arrayselec = null ; // Variable pour stocker l'élément correspondant
    var found = false; // Variable pour indiquer si l'élément correspondant a été trouvé

    // Afficher une notification "Recherche en cours"
    var notification = document.getElementById("notification");
    notification.style.display = "block";
```

Quand on clique sur un lien dans cette partie, il appelle la méthode afficherPopup dans le but d'afficher des informations détaillées de la source.

Pour cela on trouve l'api avec la clé et l'userid mais aussi une notification en haut à droite de l'écran pour informer l'utilisateur que la recherche est en cours.

```
// Démarrer la recherche récursive
var start = 0;
var query = choix; // Utilisez le choix comme critère de recherche
recursiveSearch(query, start);
```

La fonction démarre avec start et query à 0 et appelle la méthode recursiveSearch avec ces paramètres.

```

// Fonction récursive pour effectuer des recherches successives
function recursiveSearch(query, start) {
    makeSearchRequest(query, start)
        .then(function (response) {
            if (arrayselec == null) {
                if (response && response.length > 0) {
                    // S'il y a des éléments dans la réponse, continuez la recherche avec le prochain lot de résultats
                    start += 25;
                    recursiveSearch(query, start);
                } else {
                    // S'il n'y a plus de résultats, vérifiez les données et affichez la popup
                    //console.log(Apidata); // Afficher les données dans la console
                    checkData();
                }
            } else {
                checkData();
            }
        })
        .catch(function (error) {
            console.error("Erreur :", error);
            checkData(); // En cas d'erreur, vérifiez quand même les données
        });
}

```

La fonction quant à elle va appeler la fonction makeSearchRequest avec les paramètres qu'elle a reçus. Puis à chaque résultat de cette fonction, elle va tester si la demande a été trouvée ou si il reste des éléments à rechercher dans l'api. Si il en reste, elle se rappelle avec le début de la recherche +25 éléments. Dans le cas où il n'y a plus d'élément ou il a été trouvé, elle appelle la fonction checkData.

```

function makeSearchRequest(query, start) {
    var url = `https://api.zotero.org/users/${userId}/items?limit=25&start=${start}`;
    //console.log('requete');

    return new Promise(function (resolve, reject) {
        $.ajax({
            url: url,
            method: 'GET',
            success: function (response) {

                for (var i = 0; i < Apidata.length && !found; i++) {
                    var data = Apidata[i];
                    for (var y = 0; y < data.length && !found; y++) {
                        var elt = data[y];
                        if (elt['data']['shortTitle'] === choix) {
                            //console.log('-----');
                            //console.log(choix);
                            //console.log(elt);
                            //console.log('-----');
                            arrayselec = elt;
                            found = true; // Définir la variable "found" sur true pour sortir des boucles
                        }
                    }
                }
                Apidata.push(response);
                resolve(response);
            },
            error: function (xhr, status, error) {
                console.error("La requête a échoué avec le code de statut : " + xhr.status);
                resolve(null);
            }
        });
    });
}

```

Cette fonction fait une requête de 25 éléments définis par la limite dans l'url de l'api et commence aux x ème éléments définis par le start reçu. Si elle trouve l'élément, elle va modifier la variable found à true ce qui va stopper les requêtes et faire l'affichage.

```

for (var i = 0; i < Apidata.length; i++) {
    var items = Apidata[i];
    for (var j = 0; j < items.length; j++) {
        var item = items[j];
        if (item.data.shortTitle === choix) {

            if(item.data.title != null){
                titre = item.data.title;
            }
            if(item.data.creators[0].lastName != null){
                nom = item.data.creators[0].lastName;
            }
            if(item.data.creators[0].firstName != null){
                prenom = item.data.creators[0].firstName;
            }
            if(item.data.publisher != null ){
                maisonEd = item.data.publisher;
            }
            if(item.data.date != null ){
                date = item.data.date;
            }
            if(item.data.place != null ){
                place = item.data.place;
            }

            arrayselec = item;
            break;
        }
    }
}

```

Quand checkData est appelée, elle remplit les champs par le résultat récupéré.

```

// Vérifier si le titre est vide
if (titre === info) {
    notification.style.display = "none";
    // Aucune référence trouvée, afficher un message d'erreur
    var popup = window.open('', '', 'width=400,height=200');
    popup.document.write(ref);
} else {
    notification.style.display = "none";
    // Afficher les détails de la référence
    var contenuPopup = '"' + titre + '"'+ de + nom + ', '+ prenom +', '+ parue + date + par +maisonEd + a +place+';
    var popup = window.open('', '', 'width=400,height=200');
    popup.document.write(contenuPopup);
}

```

Puis si le titre est défini, elle affiche un popup avec les informations de la source et retire la

notification de recherche. Mais si le titre n'est pas défini, elle fait un popup avec information non trouvée.

Statistique

Quand on clique sur statistique, 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');
    }
}
```

Langage

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 la 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 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 informations 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, 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.

Ajout Point

Dans le menu du sidebar.

```
<li><a href="/ajout point"><?= lang('sidebar.add point button') ?></a></li>
```

Lors du clic, il appelle la route /ajout_point.

```
$routes->match(['get', 'post'],'/ajout_point', [Ajout::class, 'point']);
```

La route appelle la méthode point du contrôleur Ajout.

```
public function point()
{
    $model = model(ModelFormulaire::class);

    $data = [
        'title' => $model->getRecit(),
    ];

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

    if ($session->has('is_admin') && $session->get('is_admin') === true) {
        DatabaseUtils::insertVisit('ajout_point');

        return view('resclaves/header')
            . view('resclaves/ajout_point', $data);
    } else {
        return redirect()->to('/map');
    }
}
```

La méthode point affiche la page ajout_point.

[page] | *images/sidebar_ajoup_4.png*

La page contient un formulaire avec des champs pour chaque attribut de la table point. Quand on valide le formulaire, on utilise la route Ajout/InsertPoint.

[route] | *images/sidebar_ajoup_5.png*

La route appelle la méthode InsertPoint du contrôleur Ajout.

[méthode] | *images/sidebar_ajoup_6.png*

La méthode récupère les champs du formulaire puis les insère dans la base de données.

Ajout Récit

On retrouve aussi un bouton pour ajouter un récit.

```
<li><a href="/ajout_recit"><?= lang('sidebar.add_narrative_button') ?></a></li>
```

Il appelle la route "/ajout_recit".

```
$routes->match(['get', 'post'],'/ajout_recit', [Ajout::class, 'recit']);
```

La route appelle la méthode recit du contrôleur Ajout.

```
public function recit()
{
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);
    $model2 = model(ModelCouches::class);
    $model3 = model(ModelPolygones::class);

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

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

    if ($session->has('is_admin') && $session->get('is_admin') === true) {
        DatabaseUtils::insertVisit('ajout_recit');

        return view('resclaves/header')
            . view('resclaves/ajout_recit', $data);
    } else {
        return redirect()->to('/map');
    }
}
```

La méthode va afficher la page pour ajouter un formulaire ainsi qu'un menu déroulant avec tous les polygones dans la base de données.

```

<label><?= lang('ajout_recit.location_publication') ?></label>
<input name="lieuP" id="lieuP" type="text" value=<?php if(isset($_GET['lieuP'])){echo $_GET['lieuP'];} ?>" required /><br><br>

<!--<label>Information supplémentaire :</label>
<input name="infoSup" id="infoSup" type="text" /><br><br>-->

<label><?= lang('ajout_recit.year_publication') ?></label>
<input name="dateP" id="dateP" type="date" value=<?php if(isset($_GET['dateP'])){echo $_GET['dateP'];} ?>" required /><br><br>

<label><?= lang('ajout_recit.type_narrative') ?></label>
<input name="typeR" id="typeR" type="text" value=<?php if(isset($_GET['typeR'])){echo $_GET['typeR'];} ?>" required /><br><br>

<label><?= lang('ajout_recit.comments') ?></label>
<input name="com" id="com" type="text" value=<?php if(isset($_GET['com'])){echo $_GET['com'];} ?>" required /><br><br>

<label><?= lang('ajout_recit.method_publication') ?></label>
<input name="modeP" id="modeP" type="text" value=<?php if(isset($_GET['modeP'])){echo $_GET['modeP'];} ?>" required /><br><br>

<!--<label>Date de naissance :</label>
<input name="dateN" id="dateN" type="date" /><br><br> -->

<label><?= lang('ajout_recit.name_writer') ?></label>
<input name="nomS" id="nomS" type="text" value=<?php if(isset($_GET['nomS'])){echo $_GET['nomS'];} ?>" required /><br><br>

<label><?= lang('ajout_recit.link_narrative') ?></label>
<input name="lienR" id="lienR" type="text" value=<?php if(isset($_GET['lienR'])){echo $_GET['lienR'];} ?>" /><br><br>
```

Puis la page va avoir un formulaire à remplir.

```

<label><?= lang('ajout_recit.choix_poly') ?></label>
<select name="poly[]" id="poly" multiple required>
    <?php
        $write = true;
        if (!empty($polys) && is_array($polys)) {
            foreach ($polys as $elt) {
                if(isset($_GET['polys'])){
                    $polygones = explode(", ", $_GET['polys']);
                    $write = true;
                    for($i=0;$i<count($polygones);$i++){
                        if($polygones[$i] == $elt['name']){
                            echo '<option value="' . $elt['id'] . '" selected>' . $elt['name'] . ' </option>';
                            $write = false;
                            break;
                        }
                    }
                }
                if($write){
                    if($write){
                        echo '<option value="' . $elt['id'] . '">' . $elt['name'] . ' </option>';
                    }
                }
            }
        }
    ?>
</select><br><br>
<a class="retour" href="= site_url('/map') ?&gt;"<?= lang('recits.bouton_retour') ?></a></p>

<button type="submit"><?= lang('ajout_recit.add_button') ?></button>
```

Et un menu déroulant avec tous les polygones de la base de données.

```
$routes->post('Ajout/InsertPoly_Recit', 'Ajout::InsertPoly_Recit');
```

Quand on valide le formulaire, il va exécuter la route InsertPoly_Recit.

```

public function InsertPoly_Recit()
{
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);
    $model2 = model(ModelCouches::class);
    $model3 = model(ModelPolygones::class);
    $data = [
        'title' => $model->getRecit(),
        'auteurs' => $model1->getAuteurs(),
        'nomR' => $this->request->getPost('nomR'),
        'idE' => $this->request->getPost('idE'),
        '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'),
        'dateN' => $this->request->getPost('dateN'),
        'nomS' => $this->request->getPost('nomS'),
        'lienR' => $this->request->getPost('lienR'),
        'polys' => $this->request->getPost('poly'),
        'polygones' => $model3->getPoly()
    ];

    $idR = 0;
    foreach ($data['title'] as $elt) {
        if($elt['id_recit'] > $idR){
            $idR = $elt['id_recit'];
        }
    }
    $idR++;

    $idE = $this->request->getPost('idE');
}

```

La méthode va récupérer les informations du formulaire.

```

$idE = $this->request->getPost('idE');

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

$data += [
    'idR' => $idR,
    'nomE' => $nomE
];

return view('resclaves/header')
. view('resclaves/insert_polys', $data);

```

Puis rechercher l'id du récit et va retourner vers la page insert_poly, mais sans l'afficher.

Cette page va juste rassembler les informations pour l'appel de la méthode InsertRecit.

```

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

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);

```

Cette méthode va insérer dans la base de données le récit ainsi que les liaisons entre le récit et les polygones.

Ajout Polygone

```
<li><a href="/ajout_poly"><?= lang('sidebar.add polygon button') ?></a></li>
```

Quand on clique sur le bouton ajout polygone, il appelle la route "/ajout_poly".

```
$routes->match(['get', 'post'], '/ajout_poly', [Ajout::class, 'add_poly']);
```

La route renvoie vers la méthode add_poly du contrôleur Ajout.

```

public function add_poly (){
    return view('resclaves/header')
        . view('resclaves/ajout_polygone');
}

```

Cette méthode affiche la page ajout_polygone.

```

<form action="= site_url('Ajout/InsertPoly') ?&gt;" method="post"&gt;
    &lt;div class="input-group"&gt;
        &lt;label for="nom_poly"&gt;<?= lang('ajout_poly.poly_name') ?&gt;&lt;/label&gt;
        &lt;input type="text" id="nom_poly" name="nom_poly" required&gt;
    &lt;/div&gt;
    &lt;div class="input-group"&gt;
        &lt;div class="scrollable-table"&gt;
            &lt;table id="exa" class="display" style="width:100%;"&gt;
                &lt;thead&gt;
                    &lt;tr&gt;
                        &lt;th&gt;<?= lang('ajout_poly.point') ?&gt;&lt;/th&gt;
                        &lt;th style="position: relative; width: 30%;"&gt;<?= lang('ajout_poly.suppr') ?&gt;&lt;/th&gt;
                    &lt;/tr&gt;
                &lt;/thead&gt;
                &lt;tbody id="coordonneesTable"&gt;
                    &lt;!-- Les coordonnées seront ajoutées ici --&gt;
                &lt;/tbody&gt;
            &lt;/table&gt;
        &lt;/div&gt;
    &lt;/div&gt;

    &lt;!-- Ajoutez un champ de formulaire caché pour les coordonnées --&gt;
    &lt;input type="hidden" name="coordonnees" id="coordonneesInput"&gt;

    &lt;a class="retour" href="<?= site_url('/map') ?&gt;"<?= lang('recits.bouton_retour') ?></a></p>
    <button type="submit" id="submit-button">= lang('ajout_poly.add_poly_button') ?&gt;&lt;/button&gt;
&lt;/form&gt;
</pre

```

Cette page contient un tableau de points.

```

<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script>
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19,
}).addTo(map);

var coordonnees = []; // Tableau pour stocker les coordonnées
var polyline; // Variable pour stocker la polyline

// Fonction pour gérer les clics sur la carte
function onClick(e) {
    var latlng = e.latlng;
    var lat = latlng.lat;
    var lng = latlng.lng;

    coordonnees.push(latlng);

    // Créez une nouvelle ligne pour le tableau
    var newRow = document.createElement("tr");

    // Créez une cellule pour les coordonnées
    var coordCell = document.createElement("td");
    coordCell.textContent = lat + ", " + lng;

    // Créez une cellule pour le bouton de suppression
    var deleteCell = document.createElement("td");
    var deleteButton = document.createElement("button");
    deleteButton.textContent = "Supprimer";
    deleteButton.onclick = function() {

```

Ainsi que du JavaScript. Il y a une carte qui, quand elle est cliquée, ajoute un point dans le tableau. Si il y a plus de 2 points dans le tableau, il va dessiner un polygone.

```
$routes->post('Ajout/InsertPoly', 'Ajout::InsertPoly');
```

Puis, quand on valide le formulaire, il va chercher la méthode InsertPoly du contrôleur Ajout par la route.

```

public function InsertPoly(){

    // Récupérez les données POST
    $data = $this->request->getPost();

    // Si vous voulez vérifier ou manipuler les données reçues, vous pouvez le faire ici

    // Ensuite, $data est un tableau clé-valeur avec des clés correspondant aux noms des champs POST

    $nom_poly = $data['nom_poly'];
    $coordonnees = json_decode($data['coordonnees'], true); // Décodez la chaîne JSON en un tableau
    //var_dump($coordonnees);
    $strstart = '{"type":"MultiPolygon","coordinates":[[[';
    $strend = ']]]}';

    $geoj = $strstart;

    foreach ($coordonnees as $elt) {
        $geoj .= '[' . $elt['lat'] . ', ' . $elt['lng'] . '], ';
    }

    $geoj .= $strend;

    $sql = 'INSERT INTO `polygone` (`name`, `geoj`)
VALUES (?, ?)';
    $db = db_connect();
    $db->query($sql, [$nom_poly, $geoj]);

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

```

Cette méthode s'occupe de faire l'insertion dans la base de données du polygone créé.

Ajout Esclave/Auteur

```
<li><a href="/ajout_esclave"><?= lang('sidebar.add_slave_button') ?></a></li>
```

Quand on clique sur ajout d'un esclave, il appelle la route "/ajout_esclave".

```
$routes->match(['get', 'post'], '/ajout_esclave', [Ajout::class, 'auteur']);
```

La route appelle la méthode auteur du contrôleur Ajout.

```

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

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

    if ($session->has('is_admin') && $session->get('is_admin') === true) {
        DatabaseUtils::insertVisit('ajout_esclave');

        return view('resclaves/header')
            . view('resclaves/ajout_esclave', $data);
    } else {
        return redirect()->to('/map');
    }
}

```

La méthode affiche la page ajout_esclave.

```

<form action="= site_url('Ajout/InsertAuteur') ?&gt;" method="post"&gt;
    &lt;label&gt;&lt;?= lang('ajout_esclave.name_slave') ?&gt;&lt;/label&gt;
    &lt;input name="nomR" id="nomR" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.year_birth') ?&gt;&lt;/label&gt;
    &lt;input name="anneeN" id="anneeN" type="number" min="0" max="2030" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.location_birth') ?&gt;&lt;/label&gt;
    &lt;input name="lieuN" id="lieuN" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.year_death') ?&gt;&lt;/label&gt;
    &lt;input name="dateD" id="dateD" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.location_death') ?&gt;&lt;/label&gt;
    &lt;input name="lieuD" id="lieuD" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.location_slavery') ?&gt;&lt;/label&gt;
    &lt;input name="lieuE" id="lieuE" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.means_release') ?&gt;&lt;/label&gt;
    &lt;input name="moy" id="moy" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.location_life_after_release') ?&gt;&lt;/label&gt;
    &lt;input name="lieuV" id="lieuV" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.origin_parents') ?&gt;&lt;/label&gt;
    &lt;input name="origP" id="origP" type="text" required /&gt;&lt;br&gt;&lt;br&gt;

    &lt;label&gt;&lt;?= lang('ajout_esclave.abolitionist_activist') ?&gt;&lt;/label&gt;
    &lt;input name="mil" id="mil" type="text" required /&gt;&lt;br&gt;&lt;br&gt;
</pre

```

Dans cette page, on retrouve tous les champs pour ajouter un auteur.

```
$routes->post('Ajout/InsertAuteur', 'Ajout::InsertAuteur');
```

Quand on valide le formulaire, la route appelle la méthode InsertAuteur du contrôleur Ajout.

```

$model1 = model(ModelGetAuteur::class);
$data = [
    'title' => $model->getRecit(),
    'auteurs' => $model1->getAuteurs()
];

$nomR = $this->request->getPost('nomR');
$anneeN = $this->request->getPost('anneeN');
$lieuN = $this->request->getPost('lieuN');
$dateD = $this->request->getPost('dateD');
$lieuD = $this->request->getPost('lieuD');
$lieuE = $this->request->getPost('lieuE');
$moy = $this->request->getPost('moy');
$lieuV = $this->request->getPost('lieuV');
$origP = $this->request->getPost('origP');
$mil = $this->request->getPost('mil');
$part = $this->request->getPost('part');

$idA = 0;
foreach ($data['auteurs'] as $elt) {
    if($elt['id_auteur'] > $idA){
        $idA = $elt['id_auteur'];
    }
}
$idA++;

$plrs = "non";
$ops = "";

$sql = 'INSERT INTO `tab_auteurs` (`nom`, `naissance`, `lieu_naissance`, `deces`, `lieu_deces`, `lieu_esclavage`, `moy`
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)';
$db = db_connect();
$db->query($sql, [$nomR, $anneeN, $lieuN, $dateD, $lieuD, $lieuE, $moy, $lieuV, $origP, $mil, $part, $plrs, $idA, $ops]);

return redirect()->to('/map');

```

La méthode va insérer dans la base de données les informations remplies dans le formulaire.

Modification d'un Esclave/Auteur

```
<li><a href="/choix_esclave"><?= lang('sidebar.modify_slave_button') ?></a></li>
```

Quand on clique sur modif d'un esclave, il appelle la route "/choix_esclave".

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

La route appelle la méthode choixModifA du contrôleur Modif.

```

public function choixModifA()
{
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);

    $data = [
        'title' => $model->getRecit(),
        'auteurs' => $model1->getAuteurs()
    ];

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

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

```

La méthode récupère tous les auteurs dans la base de données et affiche la page choix_esclave.

```

<form action="= site_url('/modif_esclave') ?" method="post">

    <label><?= lang('choix_esclave.select_slave') ?></label>
    <select name="idE" id="idE" required>
        <?php
        if (!empty($auteurs) && is_array($auteurs)) {
            foreach ($auteurs as $elt) {
                echo '<option value="' . $elt['id_auteur'] . '">' . $elt['nom'] . ' </option>';
            }
        }
        ?>
    </select><br><br>

    <a class="retour" href="= site_url('/map') ?"><?= lang('recits.bouton_retour') ?></a></p>

    <button type="submit"><?= lang('choix_esclave.modify_button') ?></button>
</form>

```

La page contient un formulaire avec un menu déroulant avec tous les auteurs, ce qui permet de choisir l'auteur.

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

Le formulaire va appeler la route /modif_esclave et appeler la méthode modifA du contrôleur Modif.

```

public function modifA()
{
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);

    $data = [
        'title' => $model->getRecit(),
        'auteur' => $this->request->getPost('idE'),
        'auteurs' => $model1->getAuteurs()
    ];

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

    if ($session->has('is_admin') && $session->get('is_admin') === true) {
        if(isset($_POST['idE'])){
            return view('resclaves/header')
                . view('resclaves/modif_esclave', $data);
        } else {
            return redirect()->to('/choix_esclave');
        }
    } else {
        return redirect()->to('/map');
    }
}

```

Cette méthode va récupérer toutes les informations liées à un auteur et afficher la page modif_esclave avec les informations récupérées.

```

<form action="= site_url('Modif/ModifAuteur') ?&gt;" method="post"&gt;
    &lt;label&gt;&lt;?= lang('modif_esclave.name_slave') ?&gt;&lt;/label&gt;
    &lt;?php
    if (!empty($auteurs) &amp;&amp; is_array($auteurs)) {
        foreach ($auteurs as $elt) {
            if ($elt['id_auteur'] == $auteur) {
                echo '&lt;input name="nomE" id="nomE" type="text" value="" . $elt["nom"] . "' required/&gt;&lt;br&gt;&lt;br&gt;';
            }
        }
    }
    ?&gt;

    &lt;label&gt;&lt;?= lang('modif_esclave.year_birth') ?&gt;&lt;/label&gt;
    &lt;?php
    if (!empty($auteurs) &amp;&amp; is_array($auteurs)) {
        foreach ($auteurs as $elt) {
            if ($elt['id_auteur'] == $auteur) {
                echo '&lt;input name="anneeN" id="anneeN" type="text" value="" . $elt["naissance"] . "' required/&gt;&lt;br&gt;&lt;br&gt;';
            }
        }
    }
    ?&gt;

    &lt;label&gt;&lt;?= lang('modif_esclave.location_birth') ?&gt;&lt;/label&gt;
    &lt;?php
    if (!empty($auteurs) &amp;&amp; is_array($auteurs)) {
        foreach ($auteurs as $elt) {
            if ($elt['id_auteur'] == $auteur) {
                echo '&lt;input name="lieuN" id="lieuN" type="text" value="" . $elt["lieu_naissance"] . "' required/&gt;&lt;br&gt;&lt;br&gt;';
            }
        }
    }
    ?&gt;
</pre

```

Cette page contient un formulaire avec tous les champs à remplir pour la base de données, pré-remplis avec les informations récupérées.

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

Quand le formulaire est validé, il va appeler la méthode ModifAuteur du contrôleur Modif.

```

$nomE = $this->request->getPost('nomE');
$anneeN = $this->request->getPost('anneeN');
$lieuN = $this->request->getPost('lieuN');
$dateD = $this->request->getPost('dateD');
$lieuD = $this->request->getPost('lieuD');
$lieuE = $this->request->getPost('lieuE');
$moy = $this->request->getPost('moy');
$lieuV = $this->request->getPost('lieuV');
$origP = $this->request->getPost('origP');
$mil = $this->request->getPost('mil');
$part = $this->request->getPost('part');
$idE = $this->request->getPost('idE');
$plrs = 'non';
$opSource = 'non';
$idR = 0;

foreach($data['title'] as $elt){
    if($elt['id_auteur'] == $idE){
        $idR = $elt['id_recit'];
        $sql = 'UPDATE `tab_recits_v3` SET `nom_esc` = ? WHERE `id_recit` = ?';
        $db = db_connect();
        $db->query($sql, [$nomE, $idR]);
    }
}

$sql = 'UPDATE `tab_auteurs` SET `nom` = ?, `naissance` = ?, `lieu_naissance` = ?, `deces` = ?, `lieu_deces` = ?, `lieu_esclavage` = ?';
$db = db_connect();
$db->query($sql, [$nomE, $anneeN, $lieuN, $dateD, $lieuD, $lieuE, $moy, $lieuV, $origP, $mil, $part, $plrs, $idE, $opSource, $idR]);

```

Cette méthode va mettre à jour l'auteur.

Suppression d'un Esclave/Auteur

```
<li><a href="/suppr_esclave"><?= lang('sidebar.delete_slave_button') ?></a></li>
```

Quand on clique sur suppression d'un esclave, il appelle la route "/suppr_esclave".

```
$routes->match(['get', 'post'], '/suppr_esclave', [Suppr::class, 'supprA']);
```

La route appelle la méthode supprA du contrôleur Suppr.

```
public function supprA()
{
    $model = model(ModelFormulaire::class);
    $model1 = model(ModelGetAuteur::class);

    $data = [
        'title' => $model->getRecit(),
        'auteurs' => $model1->getAuteurs()
    ];

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

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

La méthode supprA récupère tous les auteurs dans la base de données et affiche la page suppr_esclave.

```
<form action=<?= site_url('suppr/SupprAuteur') ?>" method="post">

<label><?= lang('suppr_esclave.select_slave') ?></label>
<select name="idE" id="idE" required>
    <?php
    if (!empty($auteurs) && is_array($auteurs)) {
        foreach ($auteurs as $elt) {
            echo '<option value="' . $elt['id_auteur'] . '">' . $elt['nom'] . '</option>';
        }
    }
    ?>
</select><br><br>

<a class="retour" href=<?= site_url('/map') ?>"><?= lang('recits.bouton_retour') ?></a></p>
<button type="submit"
    onclick="return confirm('<?= lang('suppr_esclave.delete_confirmation') ?>')"><?= lang('suppr_esclave.delete_button') ?></button>
```

La page sert à choisir l'auteur à supprimer.

```
$routes->post('Suppr/SupprAuteur', 'Suppr::SupprAuteur');
```

Une fois l'auteur sélectionné et le formulaire validé, la route appelle la méthode SupprAuteur du

contrôleur Suppr.

```
public function SupprAuteur()
{
    $model = model(ModelFormulaire::class);

    $data = [
        'title' => $model->getRecit()
    ];

    $idE = $this->request->getPost('idE');

    $idR = 0;
    foreach ($data['title'] as $elt) {
        if($elt['id_auteur'] == $idE){
            $idR = $elt['id_recit'];
            $sql = 'DELETE FROM `tab_recits_v3` WHERE `id_recit` = ?';
            $db = db_connect();
            $db->query($sql, [$idR]);

            $sql = 'DELETE FROM points WHERE `points`.`id_recit` = ? ';
            $db = db_connect();
            $db->query($sql, [$idR]);
        }
    }
    $sql = 'DELETE FROM `tab_auteurs` WHERE `id_auteur` = ?';
    $db = db_connect();
    $db->query($sql, [$idE]);

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

La méthode va supprimer l'auteur sélectionné précédemment.

Footer

A propos Contact

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" ?&gt;"= lang('footer_resc.about') ?&gt;&lt;/a&gt;
    &lt;a id="lienfoot" href="<?= site_url()."contact" ?&gt;"= lang('footer_resc.contact') ?&gt;&lt;/a&gt;
  &lt;/div&gt;
&lt;/footer&gt;

&lt;style&gt;
  footer{
    background-color:#a0512db9;
  }
&lt;/style&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre

```

Contacts

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>

```

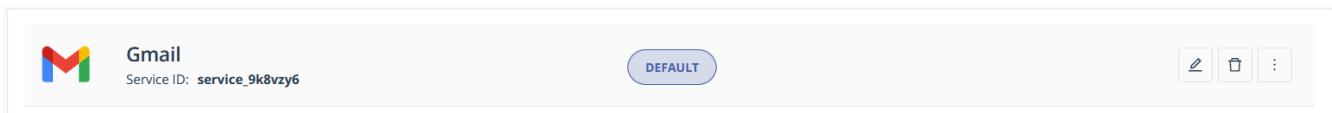
Mais on trouve aussi du JavaScript pour faire l'envoi du mail.

```

<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.

```

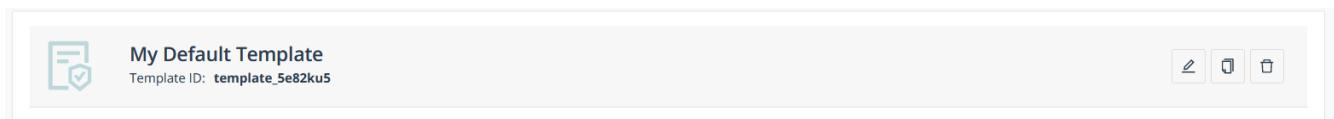
<script>
    function sendMail() {
        // Récupérer la valeur de l'e-mail
        var email = document.getElementById("email").value;

        // 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.