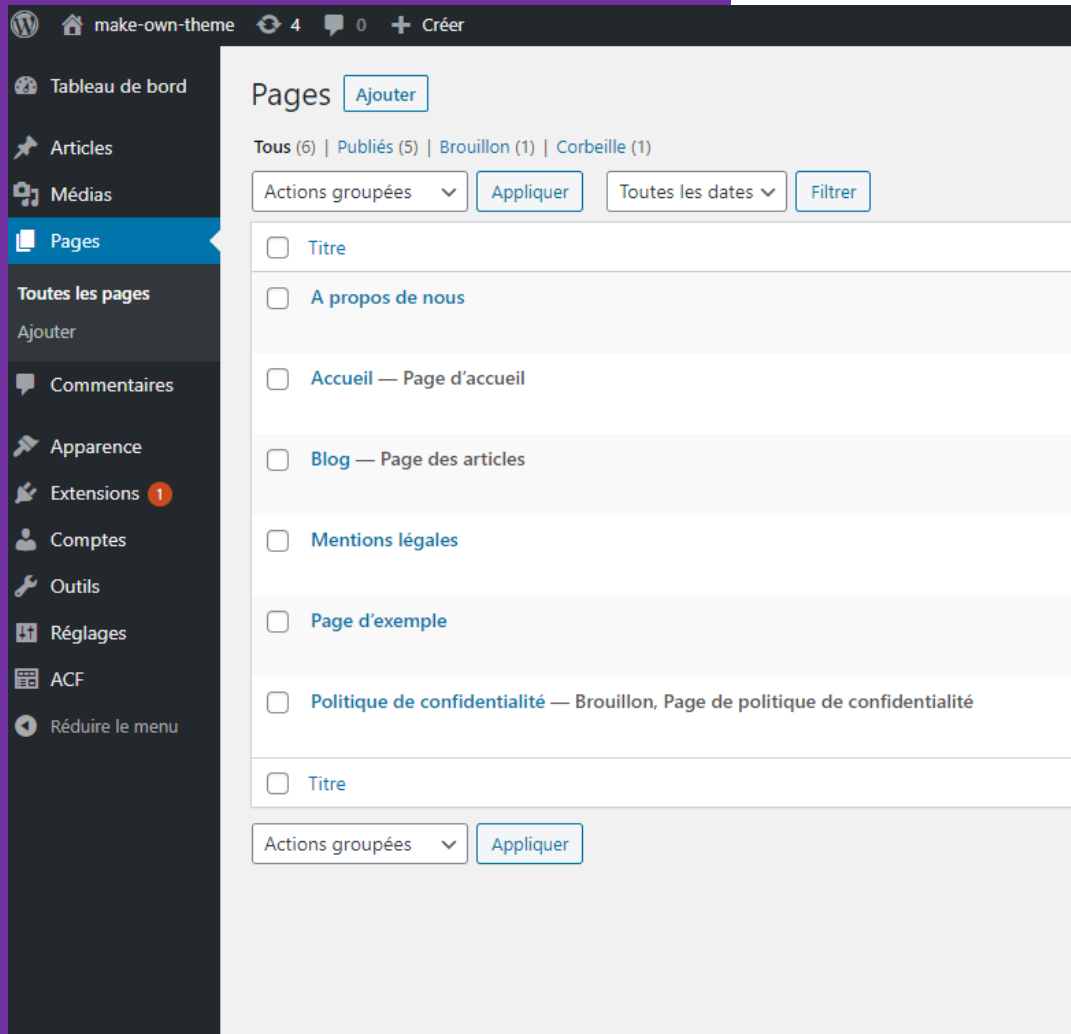




Wordpress

Création de la page des articles (single.php)

Paramétrages avant conception de la page des articles



Avant de prévoir la mise en place de la page qui servira de « template » pour les articles, il faut créer une page qui regroupera tous les articles du blog.

Sur le backoffice, il faut aller sur « Pages » puis « ajouter » pour créer une nouvelle page.

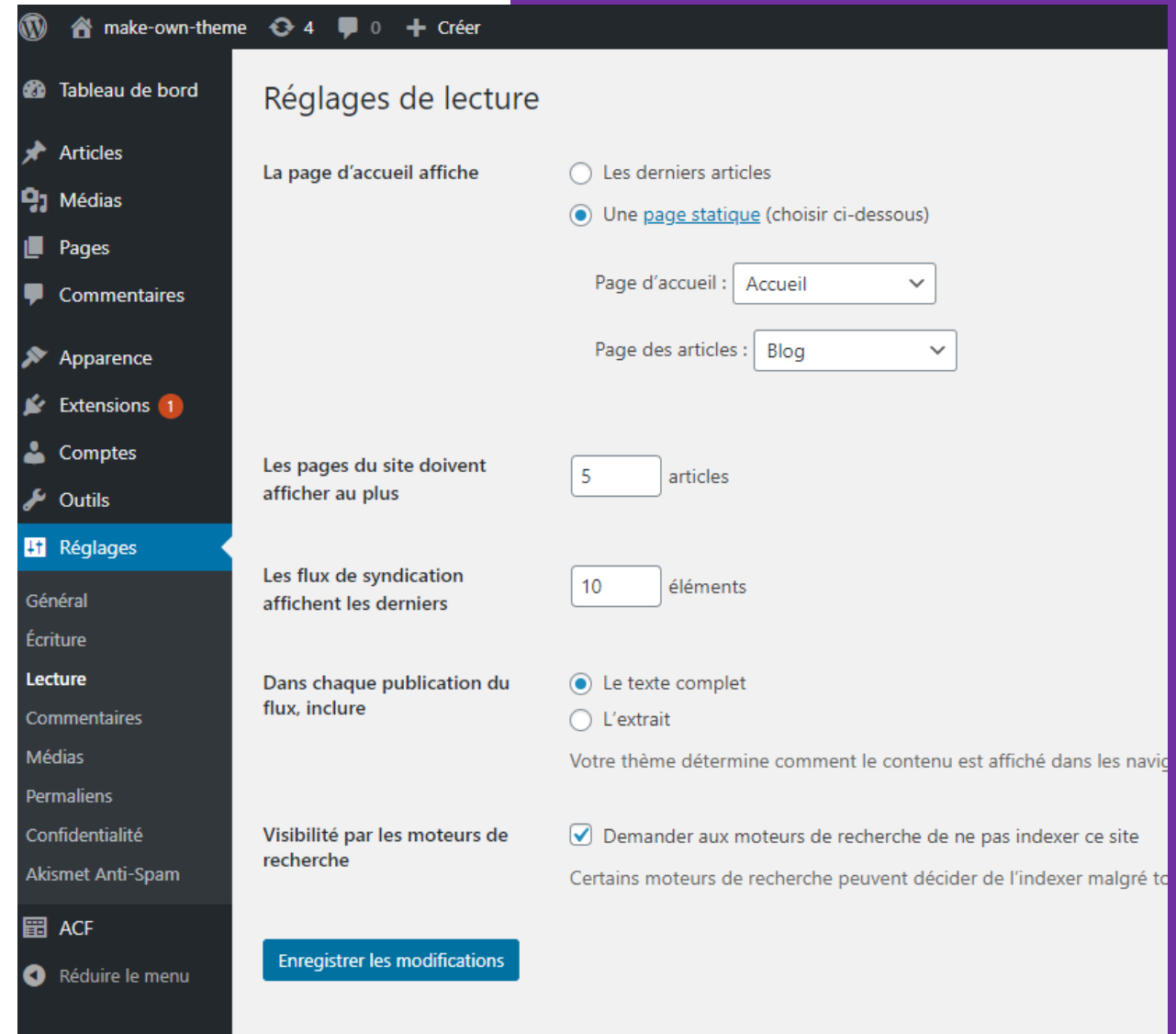
Cette nouvelle page sera nommée « blog ».

Paramétrages avant conception de la page des articles

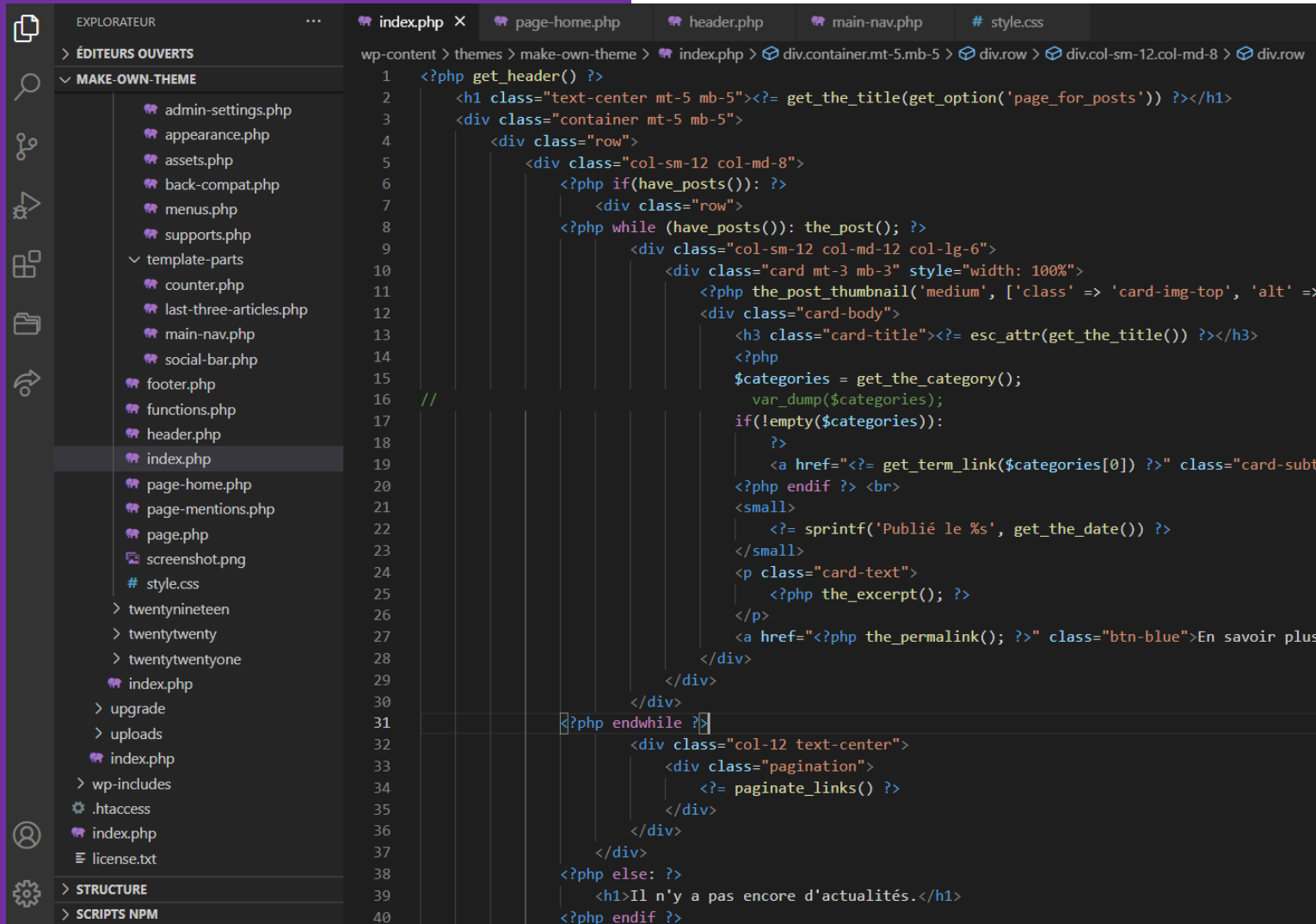
La page « blog » étant créée, il faut se rendre dans « réglages » puis « lecture » pour paramétrer les pages qui seront utilisées pour le site.

En choisissant « page statique », il est possible de choisir la page qui servira d'accueil et la page qui servira pour les articles.

Normalement, « blog » doit apparaître dans les choix possible pour la page des articles. Il suffit de laisser sur « blog » comme choix d'affichage pour la page des articles et d'enregistrer les modifications pour permettre à Wordpress d'aller chercher la bonne page.



Page pour le blog avec “index.php”



```
1 <?php get_header() ?>
2 <h1 class="text-center mt-5 mb-5"><?= get_the_title(get_option('page_for_posts')) ?></h1>
3 <div class="container mt-5 mb-5">
4   <div class="row">
5     <div class="col-sm-12 col-md-8">
6       <?php if(have_posts()): ?>
7         <div class="row">
8           <?php while (have_posts()): the_post(); ?>
9             <div class="col-sm-12 col-md-12 col-lg-6">
10              <div class="card mt-3 mb-3" style="width: 100%">
11                <?php the_post_thumbnail('medium', ['class' => 'card-img-top', 'alt' =>
12                  <div class="card-body">
13                    <h3 class="card-title"><?= esc_attr(get_the_title()) ?></h3>
14                    <?php
15                      $categories = get_the_category();
16                      var_dump($categories);
17                    if(!empty($categories)):
18                      ?>
19                      <a href="<?= get_term_link($categories[0]) ?>" class="card-subt
20                    <?php endif ?> <br>
21                    <small>
22                      <?= sprintf('Publié le %s', get_the_date()) ?>
23                    </small>
24                    <p class="card-text">
25                      <?php the_excerpt(); ?>
26                    </p>
27                    <a href="<?php the_permalink(); ?>" class="btn-blue">En savoir plus
28                  </div>
29                </div>
30              </div>
31            <?php endwhile ?>
32            <div class="col-12 text-center">
33              <div class="pagination">
34                <?= paginate_links() ?>
35              </div>
36            </div>
37          </div>
38        <?php else: ?>
39          <h1>Il n'y a pas encore d'actualités.</h1>
40        <?php endif ?>
```

Désormais c'est sur la page
« index.php » que Wordpress ira
chercher la page pour afficher les
articles du blog.

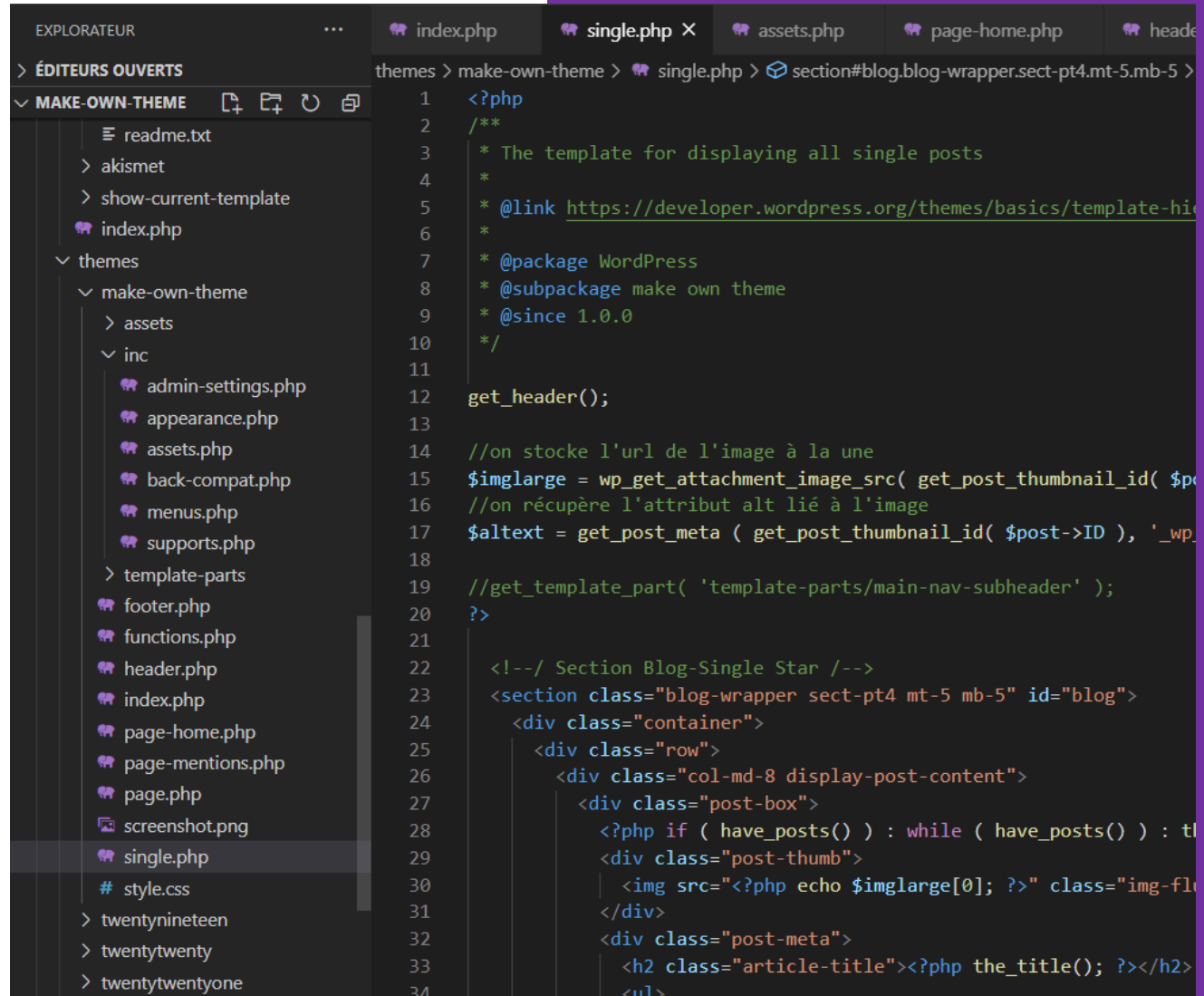
C'est donc dans ce fichier qu'il
faudra gérer l'affichage des articles
grâce à un système de boucle.

Pour afficher un article au complet, il
faut mettre en place le fichier
« single.php ».

Page d'un article complet avec "single.php"

Le fichier « single.php » est à mettre en place au même niveau qu' « index.php ».

C'est ce fichier qui affichera l'article au complet, un formulaire pour poster un commentaire mais aussi les commentaires laissés par les utilisateurs.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the following structure:

- EXPLOREUR
 - ÉDITEURS OUVERTS
 - MAKE-OWN-THEME
 - readme.txt
 - akismet
 - show-current-template
 - index.php
 - themes
 - make-own-theme
 - assets
 - inc
 - admin-settings.php
 - appearance.php
 - assets.php
 - back-compat.php
 - menus.php
 - supports.php
 - template-parts
 - footer.php
 - functions.php
 - header.php
 - index.php
 - page-home.php
 - page-mentions.php
 - page.php
 - screenshot.png
 - single.php
 - style.css
 - twentyineteen
 - twentytwenty
 - twentytwentyone

The code editor shows the content of `single.php`:

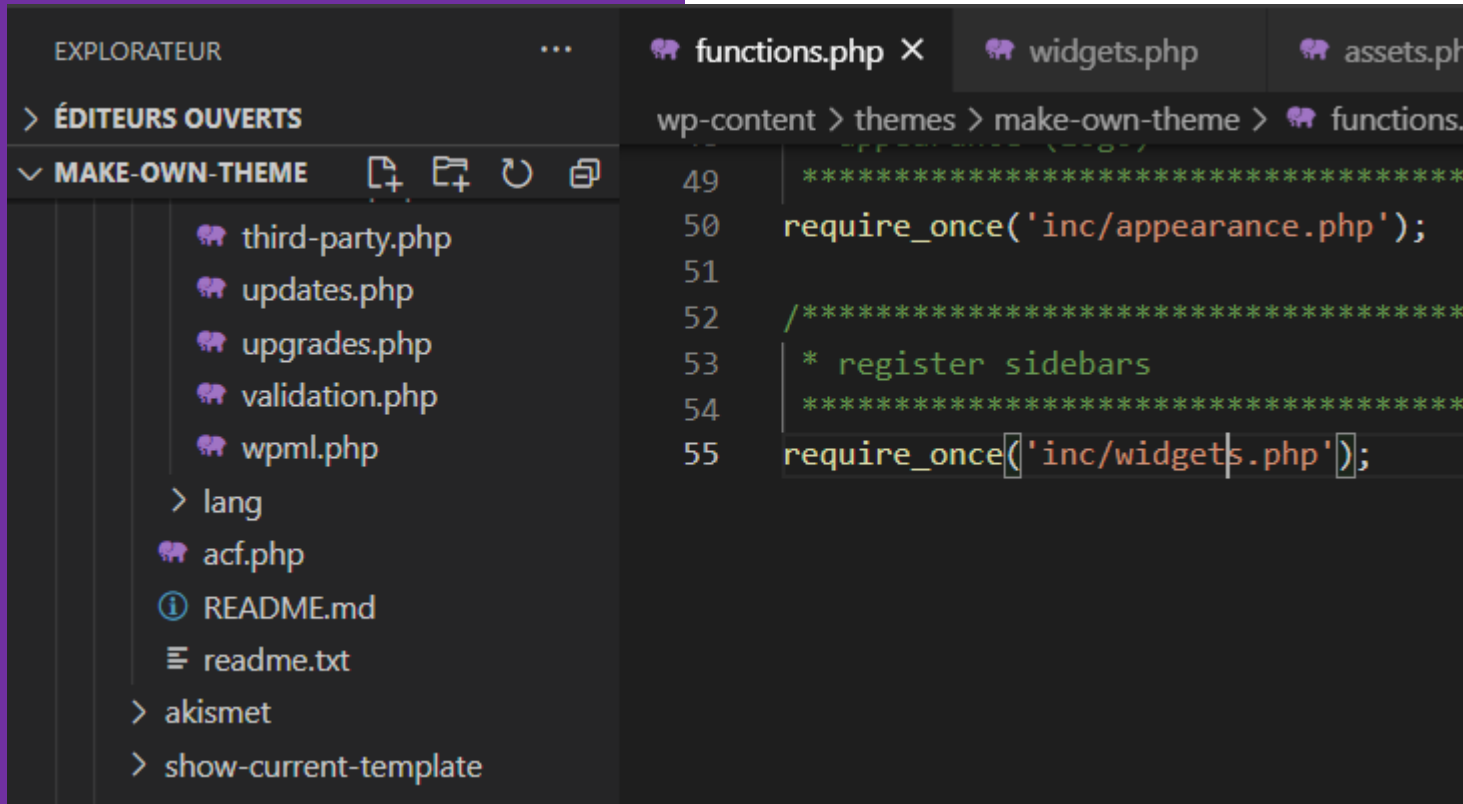
```
1 <?php
2 /**
3  * The template for displaying all single posts
4  *
5  * @link https://developer.wordpress.org/themes/basics/template-hierarchy/#single-post
6  *
7  * @package WordPress
8  * @subpackage make_own_theme
9  * @since 1.0.0
10 */
11
12 get_header();
13
14 //on stocke l'url de l'image à la une
15 $imglarge = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'full' );
16 //on récupère l'attribut alt lié à l'image
17 $alttext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
18
19 //get_template_part( 'template-parts/main-nav-subheader' );
20 ?>
21
22 <!--/ Section Blog-Single Star /-->
23 <section class="blog-wrapper sect-pt4 mt-5 mb-5" id="blog">
24     <div class="container">
25         <div class="row">
26             <div class="col-md-8 display-post-content">
27                 <div class="post-box">
28                     <?php if ( have_posts() ) : while ( have_posts() ) : the_post();
29                         <div class="post-thumb">
30                             " />
31                         </div>
32                         <div class="post-meta">
33                             <h2 class="article-title"><?php the_title(); ?></h2>
34                             <ul>
```




Wordpress

Déclaration et enregistrement d'une zone de widget pour la Sidebar

Mise en place de la sidebar dans le theme avec “functions.php”



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORATEUR' (Explorer) sidebar is open, showing the 'ÉDITEURS OUVERTS' (Open Editors) section. Under 'MAKE-OWN-THEME', a list of files is visible: third-party.php, updates.php, upgrades.php, validation.php, wpml.php, a 'lang' folder, acf.php, README.md, readme.txt, akismet, and show-current-template. The main editor area shows the 'functions.php' file. The breadcrumb path at the top reads 'wp-content > themes > make-own-theme > functions.php'. The code in the editor includes a comment on line 49, a `require_once('inc/appearance.php');` on line 50, a comment on line 52, a comment on line 53, a comment on line 54, and a `require_once('inc/widgets.php');` on line 55.

```
49  /**
50  require_once('inc/appearance.php');
51
52  /*****
53  * register sidebars
54  *****/
55  require_once('inc/widgets.php');
```

Pour tout les ajouts de support au sein de Wordpress, il est essentiel de passer par le fichier « functions.php » pour les déclarer.

Le fichier « widgets.php » qui servira pour déclarer la ou les sidebars du site.

La fonction « require_once » permet d'aller chercher le fichier nécessaire pour faire fonctionner le code.

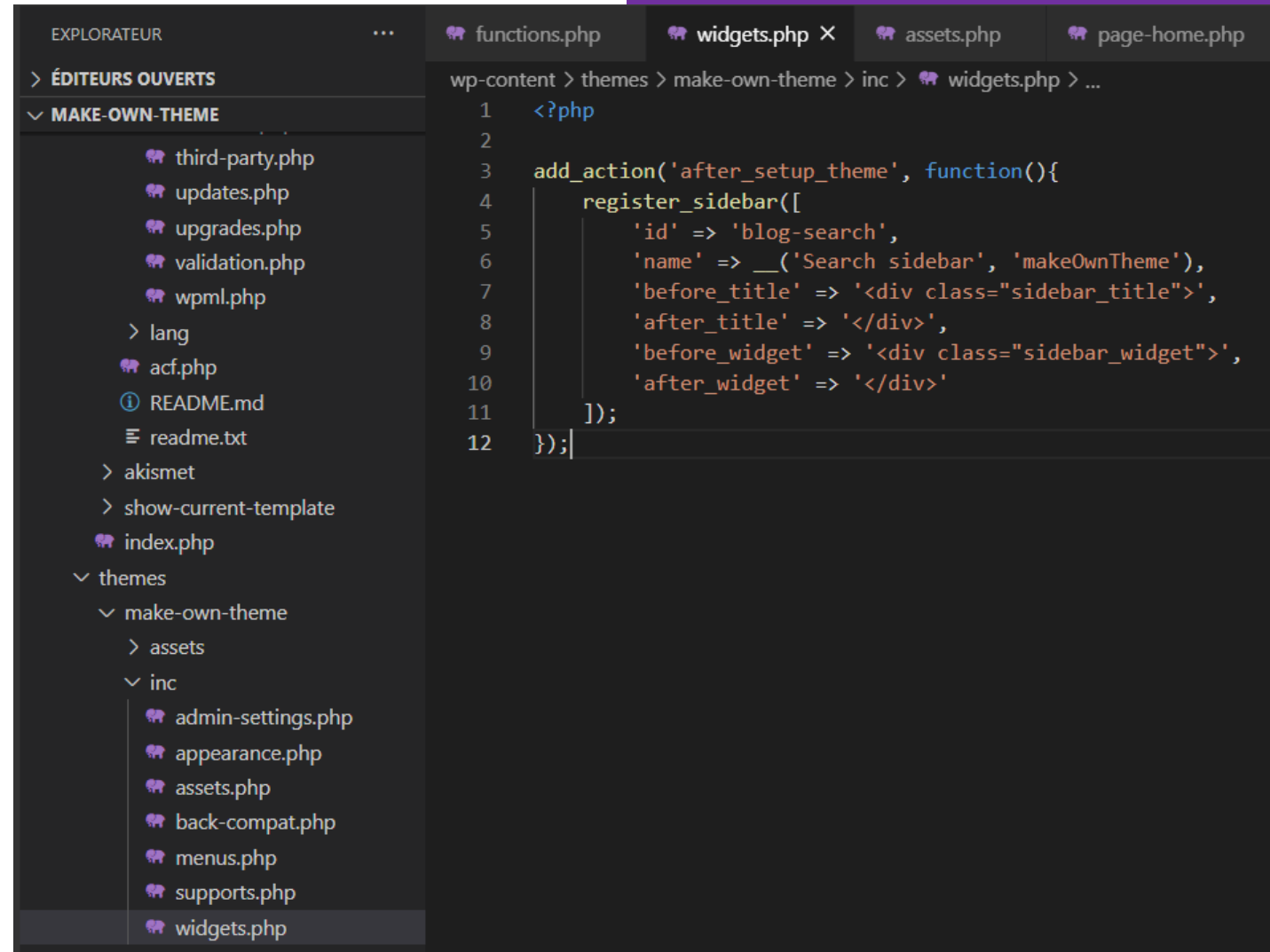
Mise en place de la sidebar dans le theme avec “widgets.php”

C'est dans le fichier « widgets.php » que va être ajouté l'appel au « hook » de Wordpress pour y greffer l'enregistrement de la sidebar.

L' « id » et le « name » s'est pour identifier la nouvelle sidebar.

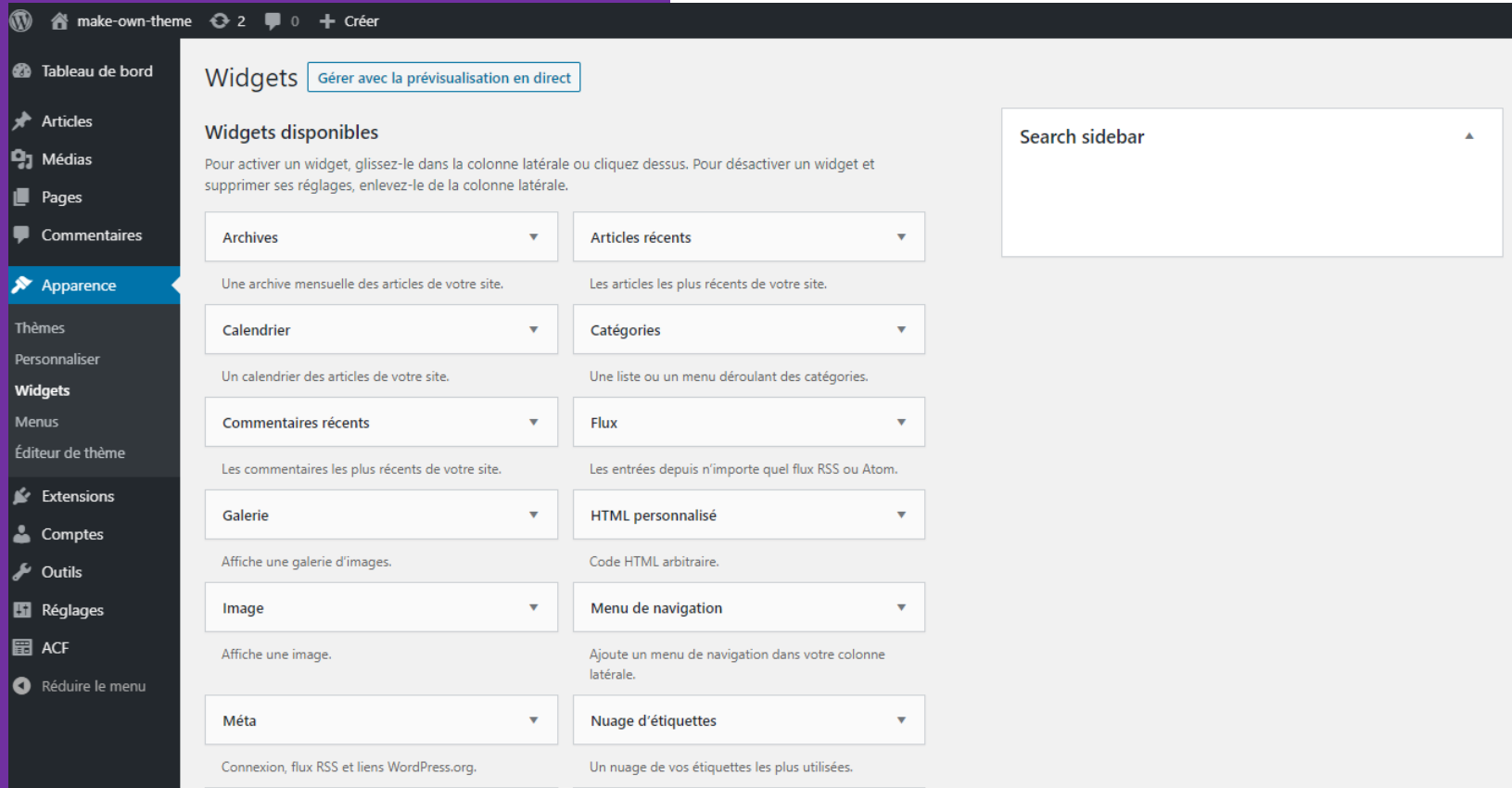
Les paramètres « before_title » et « after_title » permettent de définir l'élément HTML qui englobera la sidebar.

Les deux derniers paramètres ont la même utilité que les deux précédents mais pour les widgets qui se retrouveront dans la sidebar.



```
1 <?php
2
3 add_action('after_setup_theme', function(){
4     register_sidebar([
5         'id' => 'blog-search',
6         'name' => __('Search sidebar', 'makeOwnTheme'),
7         'before_title' => '<div class="sidebar_title">',
8         'after_title' => '</div>',
9         'before_widget' => '<div class="sidebar_widget">',
10        'after_widget' => '</div>'
11    ]);
12 });
```


Zone de widget dans le backoffice



Une fois que la sidebar est déclarée, l'onglet « widgets » doit apparaître dans le menu d'apparence.

En sélectionnant celui-ci, vous arriverez sur le menu de la capture.

A droite se trouvent les différents « widgets » disponibles.

A gauche, la sidebar qui vient juste d'être déclarée.

Il ne reste plus qu'à glisser/déposer les widgets nécessaires sur l'onglet de la sidebar.

Zone de widget, sidebar

A partir du moment où les « widgets » sont déposés dans la sidebar, un paramétrage peut être mis en place.

Comme par exemple écrire un titre ou le laisser vide.

Pour « les catégories », sélectionner plusieurs types d'affichages pour l'utilisateur.

Et pour « les articles récents » décider du nombre d'articles à afficher.

The screenshot shows the WordPress 'Widgets' management interface. The left sidebar contains the 'Apparence' menu. The main content area is titled 'Widgets' and includes a 'Gérer avec la prévisualisation en direct' button. Below this, a grid of 'Widgets disponibles' is shown, each with a description and a 'Gérer' button. To the right, three preview panels are visible:

- Search sidebar:** Shows a search box with the placeholder 'Rechercher', a 'Supprimer' link, and an 'Enregistrer' button.
- Catégories:** Shows a dropdown menu for 'Catégories', three checkboxes for display options ('Afficher comme liste déroulante', 'Afficher le nombre d'articles', 'Afficher la hiérarchie'), a 'Supprimer' link, and an 'Enregistrer' button.
- Articles récents:** Shows a title field with 'Les derniers articles', a 'Nombre d'articles à afficher' field set to '5', a checkbox for 'Afficher la date de la publication', a 'Supprimer' link, and an 'Enregistrer' button.



Wordpress

Création et inclusion du fichier sidebar.php

Sans “sidebar.php”

Sur le code qui n'est pas commenté, la fonction « dynamic_sidebar » ira chercher la sidebar qui est nommée « blog-search ».

Cette fonction a l'avantage d'aller chercher une sidebar de façon dynamique en récupérant celle par défaut qui se trouve dans le thème ou en allant récupérer une en particulier si elle est spécifiée dans ses paramètres.

Elle peut être très utile si il faut afficher plusieurs sidebar à différents endroits d'un site.

```
</div>  
<div class="col-sm-12 col-md-4 widgets">  
    <?php dynamic_sidebar('blog-search') ?>  
    <!-- <?php //get_sidebar('blog-search') ?> -->  
</div>
```

Avec “sidebar.php”

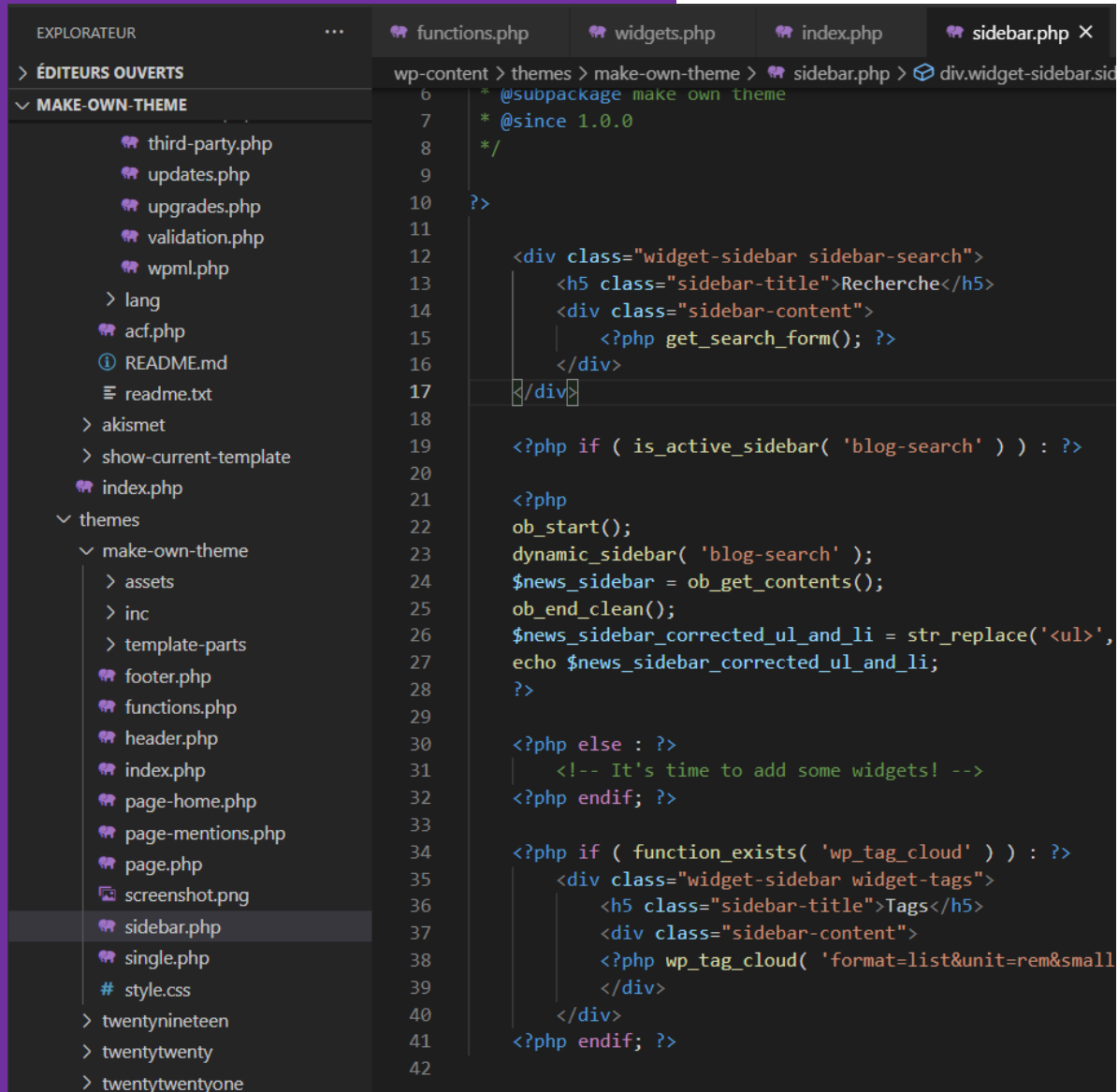
Tout comme la fonction « dynamic_sidebar », la fonction « get_sidebar » permet de récupérer une sidebar lorsqu'elle est spécifiée dans les paramètres.

Si il n'y est mis aucuns paramètres, cette fonction ira chercher le template du thème pour la sidebar par défaut.

Ce template par défaut, c'est le fichier « sidebar.php » qui est utilisé pour gérer l'affichage de la sidebar par défaut.

```
<div class="col-sm-12 col-md-4 widgets">
    <?php //dynamic_sidebar('blog-search') ?>
    <!-- <?php //get_sidebar('blog-search') ?> -->
    <?php get_sidebar(); ?>
</div>
```


Avec “sidebar.php”



```
6  * @subpackage make_own_theme
7  * @since 1.0.0
8  */
9
10 ?>
11
12 <div class="widget-sidebar sidebar-search">
13     <h5 class="sidebar-title">Recherche</h5>
14     <div class="sidebar-content">
15         <?php get_search_form(); ?>
16     </div>
17 </div>
18
19 <?php if ( is_active_sidebar( 'blog-search' ) ) : ?>
20
21 <?php
22 ob_start();
23 dynamic_sidebar( 'blog-search' );
24 $news_sidebar = ob_get_contents();
25 ob_end_clean();
26 $news_sidebar_corrected_ul_and_li = str_replace('<ul>',
27 echo $news_sidebar_corrected_ul_and_li;
28 ?>
29
30 <?php else : ?>
31     <!-- It's time to add some widgets! -->
32 <?php endif; ?>
33
34 <?php if ( function_exists( 'wp_tag_cloud' ) ) : ?>
35     <div class="widget-sidebar widget-tags">
36         <h5 class="sidebar-title">Tags</h5>
37         <div class="sidebar-content">
38             <?php wp_tag_cloud( 'format=list&unit=rem&small
39             </div>
40         </div>
41 <?php endif; ?>
42
```

Sur cette capture, voici un exemple de code qui permet de gérer l’affichage de la sidebar.

Une vérification est faite en amont pour être sûr que la sidebar est bien activée dans le back office.

Ensuite l’affichage est géré à la volée pour les « ul » et « li » pour être remplacé par une structure HTML plus claire.



Wordpress

Création et inclusion du fichier
searchform.php

searchForm.php

```
<div class="widget-sidebar sidebar-search">
  <h5 class="sidebar-title">Recherche</h5>
  <div class="sidebar-content">
    <!-- display search form -->
    <?php get_search_form(); ?>
  </div>
</div>
```

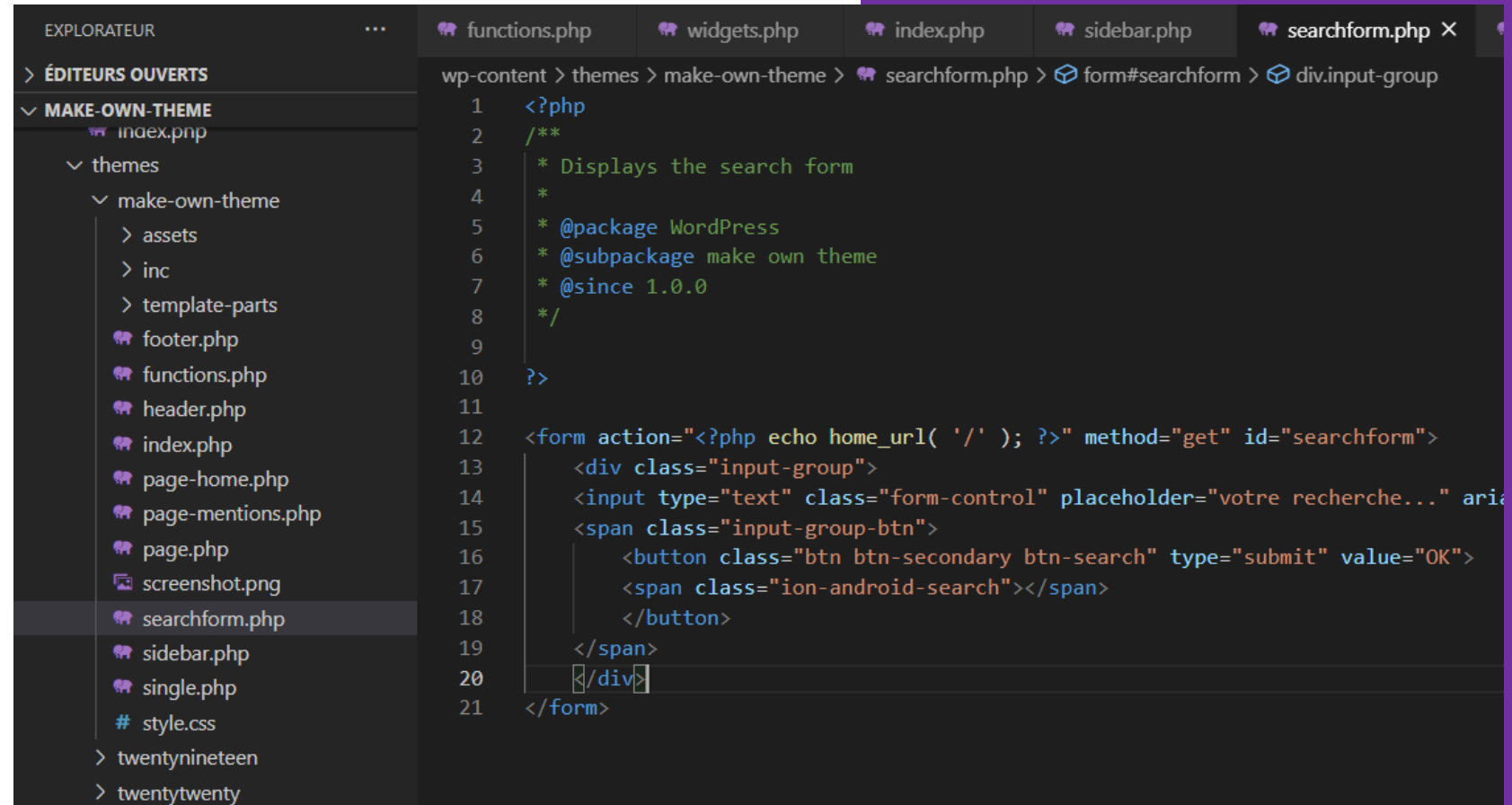
Dans la sidebar ou ailleurs sur le site, il est possible de mettre en place un formulaire de recherche.

C'est la fonction « `get_search_form` » qui se chargera de récupérer le formulaire à afficher.

Elle ira tout simplement chercher le fichier « `searchform.php` » pour gérer l'affichage.

searchForm.php

C'est dans ce fichier
« searchform.php » que le formulaire
est mis en place pour permettre à
l'utilisateur de faire sa recherche.



The screenshot shows a code editor with a sidebar on the left and a main editor area on the right. The sidebar, titled 'EXPLORATEUR', shows a file tree for a WordPress theme named 'make-own-theme'. The tree includes folders like 'assets', 'inc', and 'template-parts', and files like 'footer.php', 'functions.php', 'header.php', 'index.php', 'page-home.php', 'page-mentions.php', 'page.php', 'screenshot.png', 'searchform.php' (highlighted), 'sidebar.php', 'single.php', and 'style.css'. The main editor area shows the content of 'searchform.php'. The code is in PHP and HTML, defining a search form. It starts with a PHP opening tag and a comment block describing the file's purpose and package information. The main content is an HTML form with an action attribute that uses a PHP function to echo the home URL. The form contains a text input field with a placeholder 'votre recherche...' and a submit button with the value 'OK'. The code is as follows:

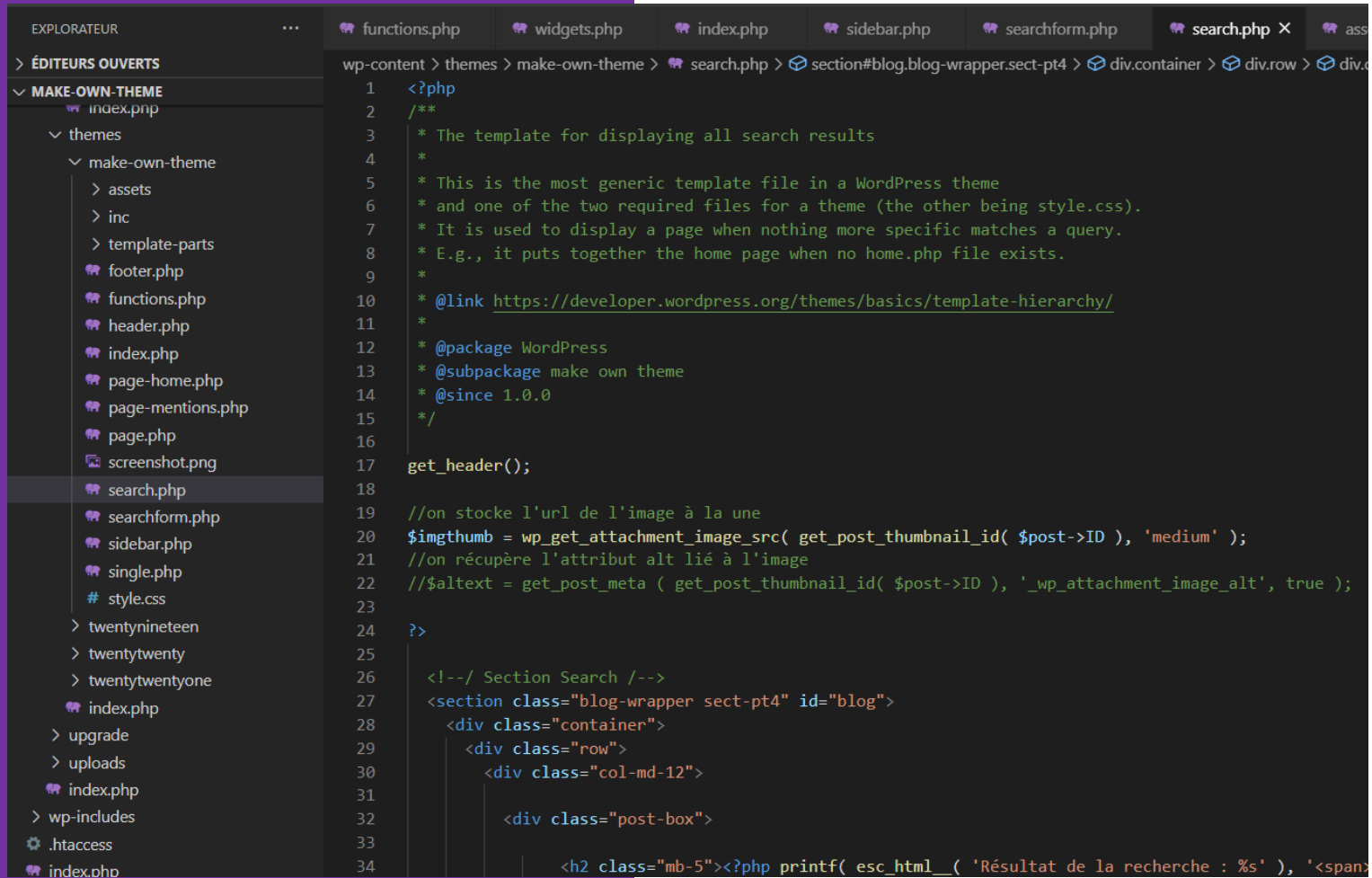
```
1  <?php
2  /**
3   * Displays the search form
4   *
5   * @package WordPress
6   * @subpackage make own theme
7   * @since 1.0.0
8   */
9
10 ?>
11
12 <form action="<?php echo home_url( '/' ); ?>" method="get" id="searchform">
13     <div class="input-group">
14         <input type="text" class="form-control" placeholder="votre recherche..." aria-label="Search">
15         <span class="input-group-btn">
16             <button class="btn btn-secondary btn-search" type="submit" value="OK">
17                 <span class="ion-android-search"></span>
18             </button>
19         </span>
20     </div>
21 </form>
```




Wordpress

Création d'un fichier search.php pour
l'affichage des résultats de recherche

Search.php



The screenshot shows a code editor with a sidebar on the left and a main editor area on the right. The sidebar, titled 'EXPLORATEUR', shows a file tree for a WordPress theme named 'make-own-theme'. The 'themes' folder is expanded, showing subfolders like 'assets', 'inc', and 'template-parts', and various PHP files including 'search.php', which is currently selected. The main editor area displays the code for 'search.php'. The code starts with a PHP opening tag and a multi-line comment explaining the file's purpose: 'The template for displaying all search results'. It includes a link to the WordPress theme hierarchy documentation, package information, and a call to 'get_header()'. Below this, there are comments and code for retrieving the image thumbnail and alt text for the post being searched. The code then uses PHP to print the search results, with a line partially visible at the bottom: '<h2 class="mb-5"><?php printf(esc_html_('Résultat de la recherche : %s'), ''. The breadcrumb navigation at the top of the editor shows the path: 'wp-content > themes > make-own-theme > search.php > section#blog.blog-wrapper.sect-pt4 > div.container > div.row > div.d'.

```
1 <?php
2 /**
3  * The template for displaying all search results
4  *
5  * This is the most generic template file in a WordPress theme
6  * and one of the two required files for a theme (the other being style.css).
7  * It is used to display a page when nothing more specific matches a query.
8  * E.g., it puts together the home page when no home.php file exists.
9  *
10 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
11 *
12 * @package WordPress
13 * @subpackage make own theme
14 * @since 1.0.0
15 */
16
17 get_header();
18
19 //on stocke l'url de l'image à la une
20 $imgthumb = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'medium' );
21 //on récupère l'attribut alt lié à l'image
22 //$alttext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
23
24 ?>
25
26 <!--/ Section Search /-->
27 <section class="blog-wrapper sect-pt4" id="blog">
28   <div class="container">
29     <div class="row">
30       <div class="col-md-12">
31
32         <div class="post-box">
33
34           <h2 class="mb-5"><?php printf( esc_html_( 'Résultat de la recherche : %s' ), '<span>
```

Pour afficher les résultats de la recherche qui ont été passé via le formulaire de recherche, il faut créer le fichier « search.php ».



Wordpress

Utilisation de la boucle avec des arguments
pour affichage dans index.php

Boucle pour affichages sur index.php

```
<?php if(have_posts()): ?>
    <div class="row">
    <?php while (have_posts()): the_post(); ?>
        <div class="col-sm-12 col-md-12 col-lg-6">
            <div class="card mt-3 mb-3" style="width: 100%">
                <?php the_post_thumbnail('medium', ['class' => 'card-img-top', 'alt' => '']); ?>
                <div class="card-body">
                    <h3 class="card-title"><? esc_attr(get_the_title()) ?></h3>
                    <?php
                    $categories = get_the_category();
                    var_dump($categories);
                    if(!empty($categories)):
                        ?>
                        <a href="<? get_term_link($categories[0]) ?>" class="card-subtitles mb-2 text-muted"><? $categories[0]
                        <?php endif ?> <br>
                        <small>
                            <?= sprintf('Publié le %s', get_the_date()) ?>
                        </small>
                        <p class="card-text">
                            <?php the_excerpt(); ?>
                        </p>
                        <a href="<?php the_permalink(); ?>" class="btn-blue">En savoir plus</a>
                    </div>
                </div>
            </div>
        </div>
    <?php endwhile ?>
    <div class="col-12 text-center">
        <div class="pagination">
            <?= paginate_links() ?>
        </div>
    </div>
<?php else: ?>
    <h1>Il n'y a pas encore d'actualités.</h1>
<?php endif ?>
```

Pour afficher les résultats d'une requête sur les articles, il faut tout d'abord utiliser la fonction « have_posts » pour vérifier si il y a des articles.

Ensuite pour afficher article par article, il faut utiliser la boucle « while » (ou « tant que » en français).

C'est donc a chaque itération de cette boucle qu'un article sera affiché.

Boucle pour affichages sur index.php

Pour chaque itération, il faut afficher l'image à la une avec « the_post_thumbnail », la catégorie avec « get_the_category », la date de publication avec « the_date », l'extrait de l'article avec « the_excerpt » et un lien vers l'article complet avec « the_permalink ».

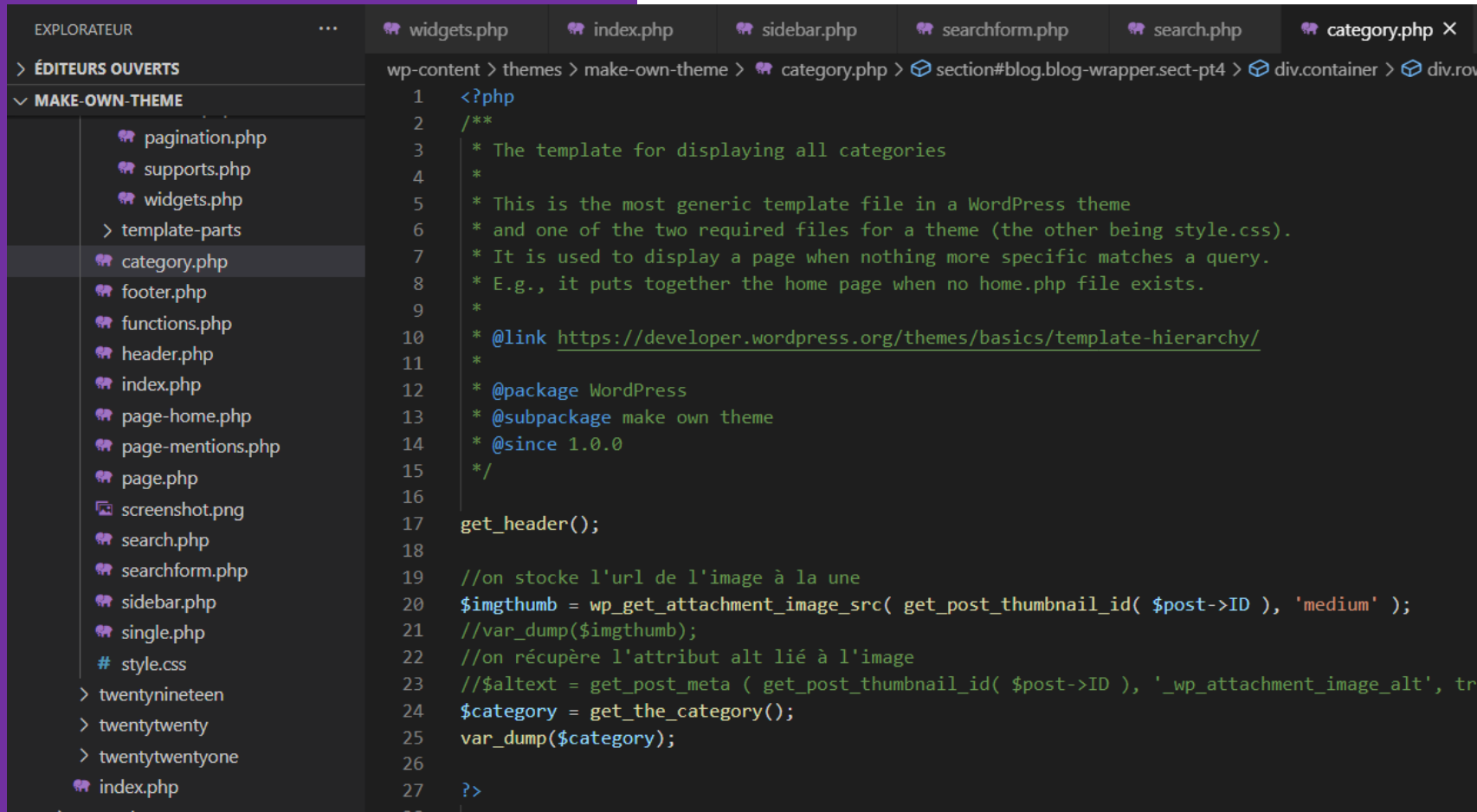
```
<?php while (have_posts()): the_post(); ?>
    <div class="col-sm-12 col-md-12 col-lg-6">
        <div class="card mt-3 mb-3" style="width: 100%">
            <?php the_post_thumbnail('medium', ['class' => 'card-img-top', 'alt' => '']); ?>
            <div class="card-body">
                <h3 class="card-title"><?= esc_attr(get_the_title()) ?></h3>
                <?php
                $categories = get_the_category();
                var_dump($categories);
                if(!empty($categories)):
                    ?>
                    <a href="<?= get_term_link($categories[0]) ?>" class="card-subtitles mb-2 text-muted"><?= $categories[0]-
                <?php endif ?> <br>
                <small>
                    <?= sprintf('Publié le %s', get_the_date()) ?>
                </small>
                <p class="card-text">
                    <?php the_excerpt(); ?>
                </p>
                <a href="<?php the_permalink(); ?>" class="btn-blue">En savoir plus</a>
            </div>
        </div>
    </div>
<?php endwhile ?>
```




Wordpress

Création de la page category.php

Category.php



The screenshot shows a code editor with a sidebar on the left and a main editor area on the right. The sidebar, titled 'EXPLORATEUR', shows a file tree for a WordPress theme named 'MAKE-OWN-THEME'. The file 'category.php' is selected. The main editor area shows the code for 'category.php', which is a template for displaying all categories. The code includes comments explaining its purpose and usage, and a PHP function call 'get_header()' at the top.

```
1 <?php
2 /**
3  * The template for displaying all categories
4  *
5  * This is the most generic template file in a WordPress theme
6  * and one of the two required files for a theme (the other being style.css).
7  * It is used to display a page when nothing more specific matches a query.
8  * E.g., it puts together the home page when no home.php file exists.
9  *
10 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
11 *
12 * @package WordPress
13 * @subpackage make own theme
14 * @since 1.0.0
15 */
16
17 get_header();
18
19 //on stocke l'url de l'image à la une
20 $imgthumb = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'medium' );
21 //var_dump($imgthumb);
22 //on récupère l'attribut alt lié à l'image
23 //$alttext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
24 $category = get_the_category();
25 var_dump($category);
26
27 ?>
```

Pour gérer l'affichage de tous les articles d'une catégorie donnée, il faut mettre en place le fichier « category.php ».

C'est dans ce fichier qu'il faut faire la requête qui va aller chercher les articles de la catégorie concernée.

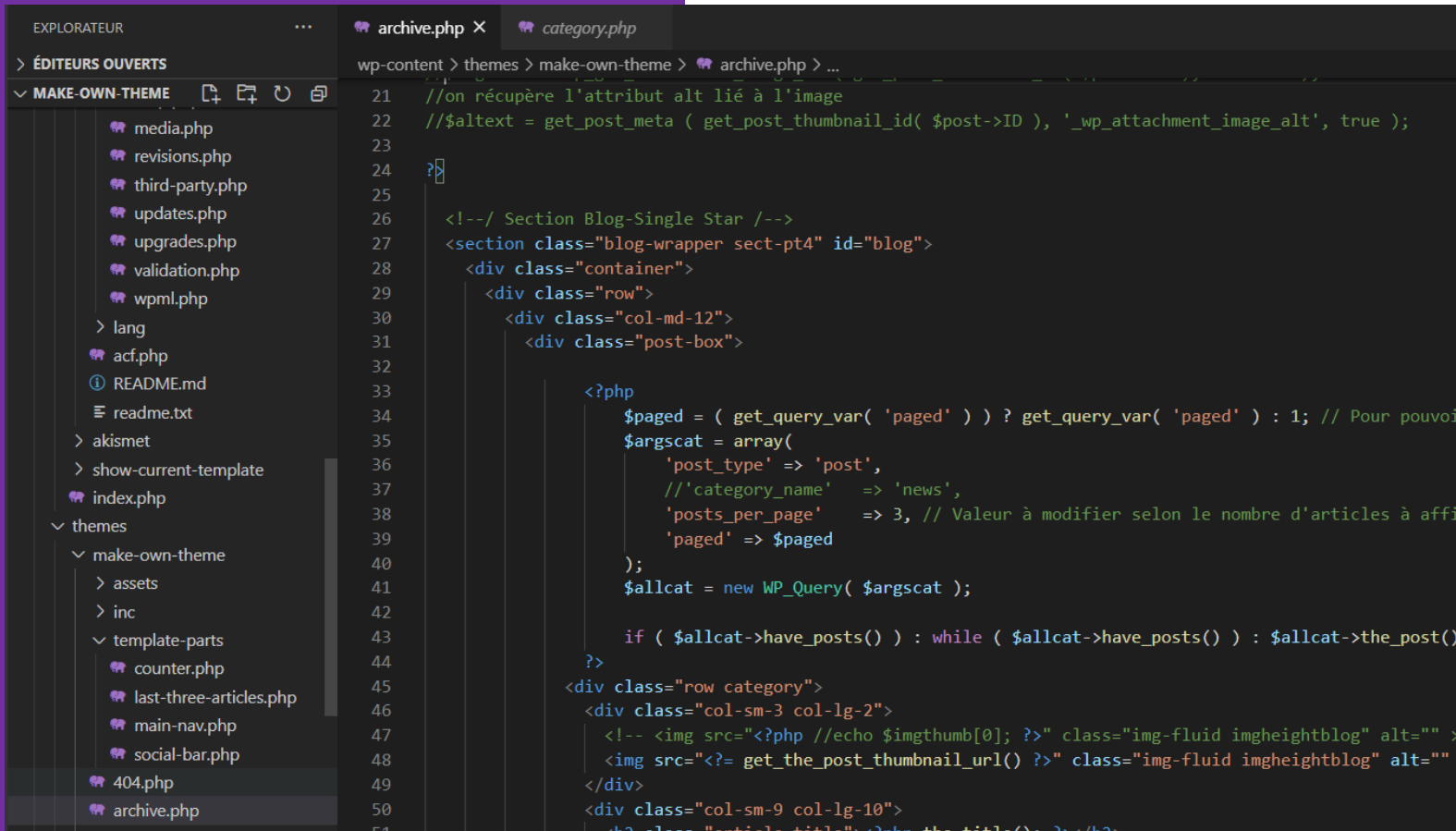
Ensuite, il ne reste plus qu'à mettre en place l'affichage si des articles existent bien dans la catégorie.



Wordpress

Création de la page archive.php

Archive.php



```
21 //on récupère l'attribut alt lié à l'image
22 // $alttext = get_post_meta( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
23
24 ?>
25
26 <!-- / Section Blog-Single Star /-->
27 <section class="blog-wrapper sect-pt4" id="blog">
28   <div class="container">
29     <div class="row">
30       <div class="col-md-12">
31         <div class="post-box">
32
33           <?php
34             $paged = ( get_query_var( 'paged' ) ) ? get_query_var( 'paged' ) : 1; // Pour pouvoir
35             $argscat = array(
36               'post_type' => 'post',
37               //'category_name' => 'news',
38               'posts_per_page' => 3, // Valeur à modifier selon le nombre d'articles à affi
39               'paged' => $paged
40             );
41             $allcat = new WP_Query( $argscat );
42
43             if ( $allcat->have_posts() ) : while ( $allcat->have_posts() ) : $allcat->the_post()
44               ?>
45               <div class="row category">
46                 <div class="col-sm-3 col-lg-2">
47                   <!-- 
48                   
50                 <div class="col-sm-9 col-lg-10">
51                   <h2 class="article-title"><?php the_title(); ?></h2>
```

La page « archive.php » comme son nom l'indique, permet d'afficher tous les articles en les archivant soit par dates et/ou catégories.

C'est dans ce fichier qu'il faut faire une requête qui va aller chercher tous les articles.

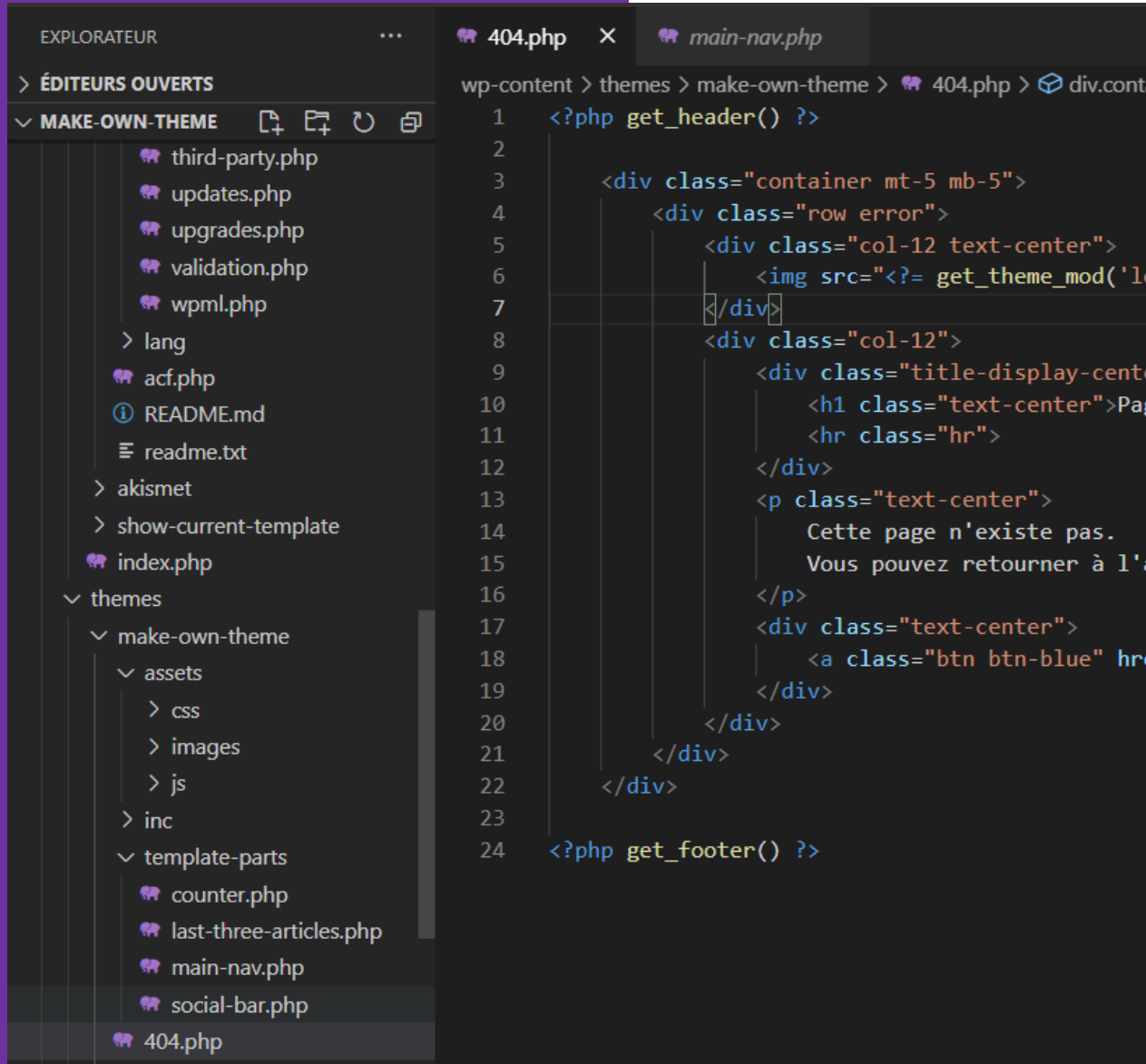
Ensuite, il ne reste plus qu'à mettre en place l'affichage si des articles existent.



Wordpress

Création de la page 404.php

404.php



The screenshot shows a code editor with a sidebar on the left displaying the file structure of a WordPress theme. The main editor area shows the content of the 404.php file. The sidebar lists files like third-party.php, updates.php, upgrades.php, validation.php, wpml.php, lang, acf.php, README.md, readme.txt, akismet, show-current-template, index.php, themes, make-own-theme, assets, css, images, js, inc, template-parts, counter.php, last-three-articles.php, main-nav.php, social-bar.php, and 404.php. The main editor shows the following code:

```
1 <?php get_header() ?>
2
3 <div class="container mt-5 mb-5">
4     <div class="row error">
5         <div class="col-12 text-center">
6             
9         <div class="title-display-cent
10             <h1 class="text-center">Pag
11             <hr class="hr">
12         </div>
13         <p class="text-center">
14             Cette page n'existe pas.
15             Vous pouvez retourner à l'a
16         </p>
17         <div class="text-center">
18             <a class="btn btn-blue" href
19         </div>
20     </div>
21 </div>
22
23
24 <?php get_footer() ?>
```

Quand un utilisateur tape une mauvaise url ou tout simplement quand il est redirigé vers une mauvaise adresse, il est essentiel de lui indiquer qu'une erreur est survenue.

En créant le fichier « 404.php », Wordpress sera quel affichage utiliser lorsqu'une url n'est pas bonne.

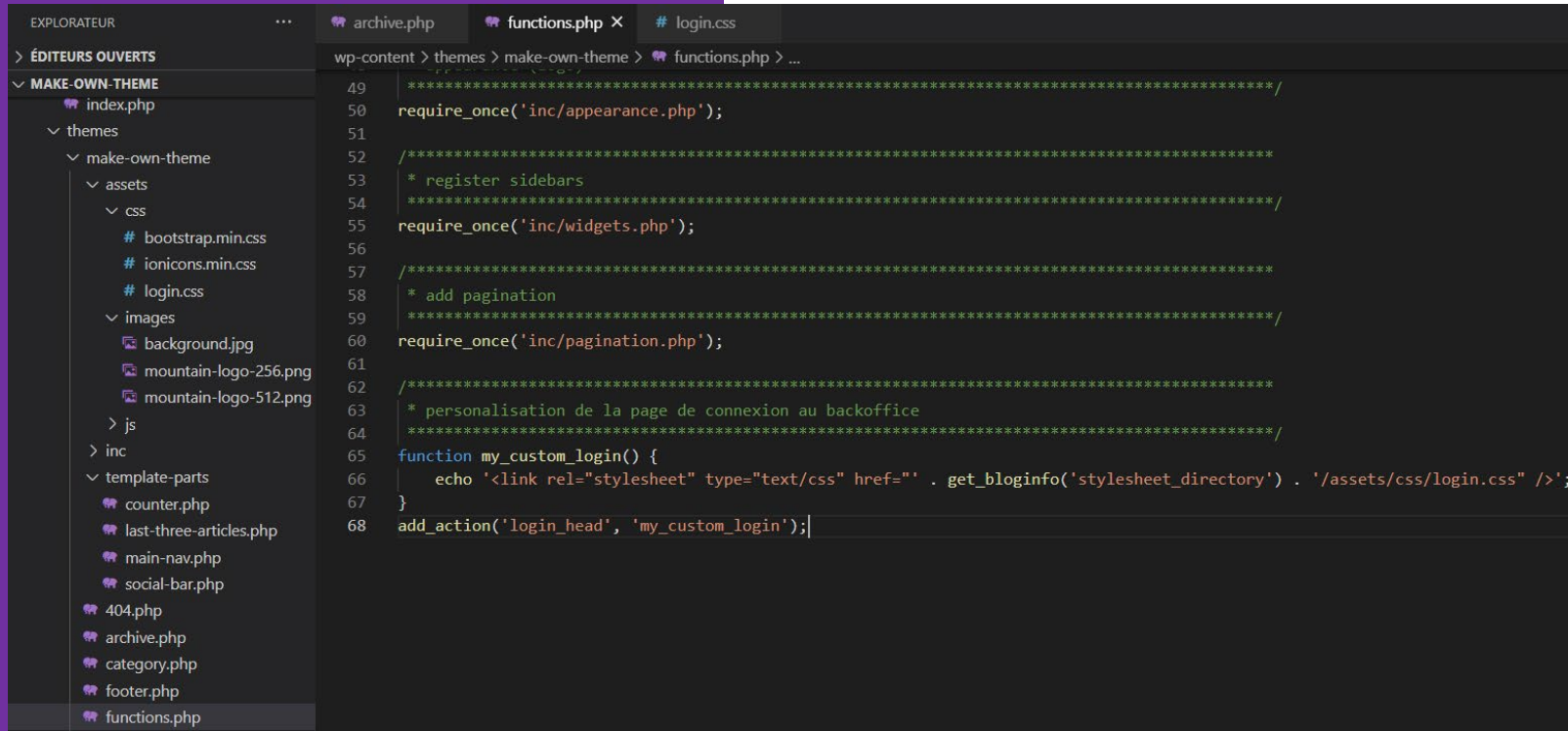
Dans ce fichier, il faut en général mettre un texte explicatif pour prévenir l'utilisateur et un bouton pour qu'il puisse revenir à la page d'accueil.



Wordpress

Personnalisation de la page login.php

Login.php



The screenshot shows a code editor with a sidebar on the left displaying the file explorer. The main editor area shows the contents of the `functions.php` file. The sidebar lists the following structure:

- EXPLOREUR
- ÉDITEURS OUVERTS
- MAKE-OWN-THEME
 - index.php
 - themes
 - make-own-theme
 - assets
 - css
 - bootstrap.min.css
 - ionicons.min.css
 - login.css
 - images
 - background.jpg
 - mountain-logo-256.png
 - mountain-logo-512.png
 - js
 - inc
 - template-parts
 - counter.php
 - last-three-articles.php
 - main-nav.php
 - social-bar.php
 - 404.php
 - archive.php
 - category.php
 - footer.php
 - functions.php

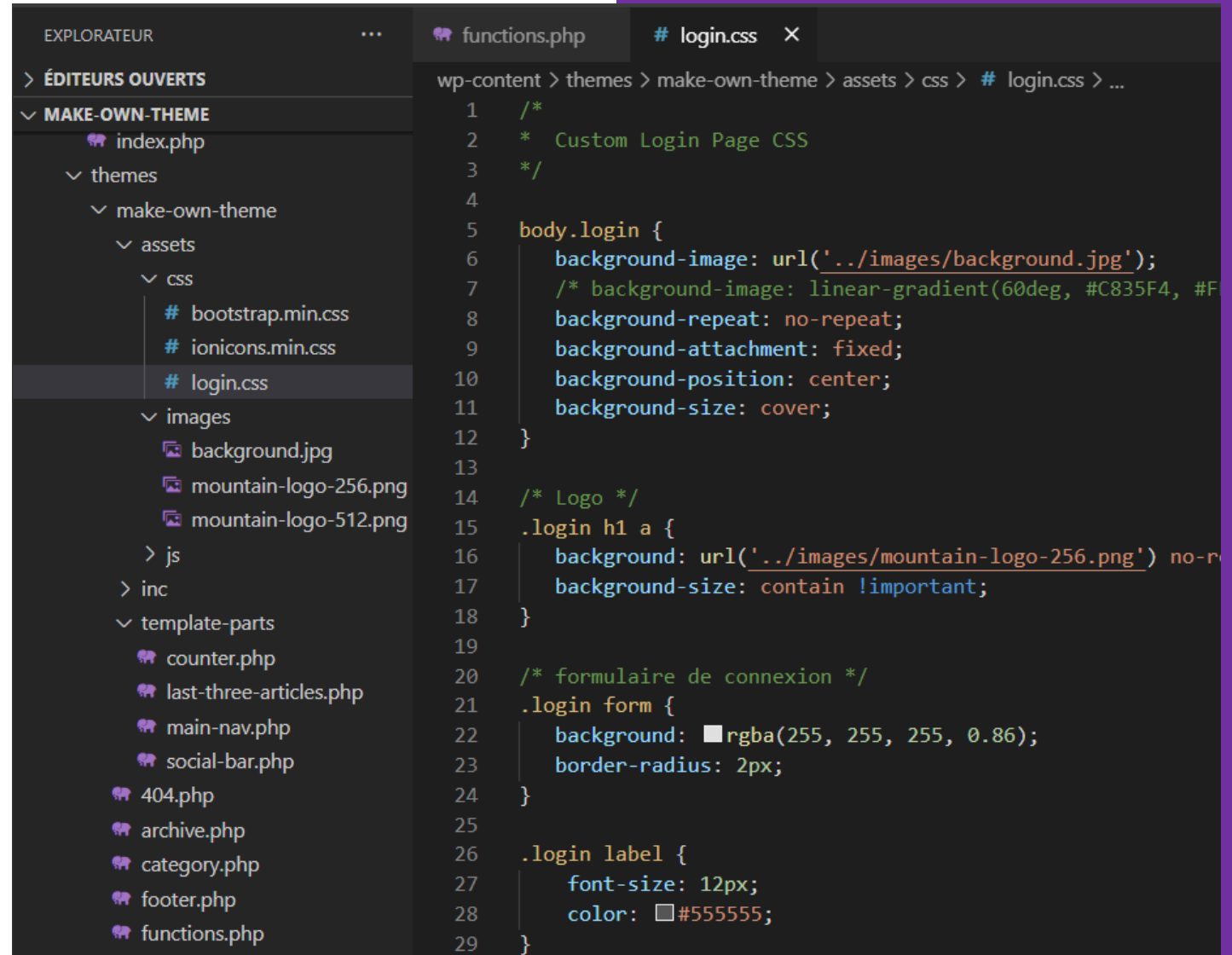
The main editor shows the following code in `functions.php`:

```
49  //*****  
50  require_once('inc/appearance.php');  
51  
52  //*****  
53  * register sidebars  
54  //*****  
55  require_once('inc/widgets.php');  
56  
57  //*****  
58  * add pagination  
59  //*****  
60  require_once('inc/pagination.php');  
61  
62  //*****  
63  * personnalisation de la page de connexion au backoffice  
64  //*****  
65  function my_custom_login() {  
66      echo '<link rel="stylesheet" type="text/css" href="' . get_bloginfo('stylesheet_directory') . '/assets/css/login.css' />';  
67  }  
68  add_action('login_head', 'my_custom_login');
```

Pour modifier la mise en forme de la page de login du backoffice propre à Wordpress. Il faut se brancher sur le « hook » de la page de connexion : « login_head » pour y ajouter le style custom que l'on veut appliquer.

Login.php

Une fois que la feuille de style est créée et liée au CMS, il ne reste plus qu'à scruter les classes utilisées sur la page de « login » du backoffice pour modifier le style de celle-ci.



The screenshot shows a code editor with two panes. The left pane, titled 'EXPLORATEUR', displays a file tree for a project named 'MAKE-OWN-THEME'. The tree structure is as follows:

- MAKE-OWN-THEME
 - index.php
 - themes
 - make-own-theme
 - assets
 - css
 - # bootstrap.min.css
 - # icons.min.css
 - # login.css (selected)
 - images
 - background.jpg
 - mountain-logo-256.png
 - mountain-logo-512.png
 - js
 - inc
 - template-parts
 - counter.php
 - last-three-articles.php
 - main-nav.php
 - social-bar.php
 - 404.php
 - archive.php
 - category.php
 - footer.php
 - functions.php

The right pane, titled '# login.css', shows the content of the selected file. The code is as follows:

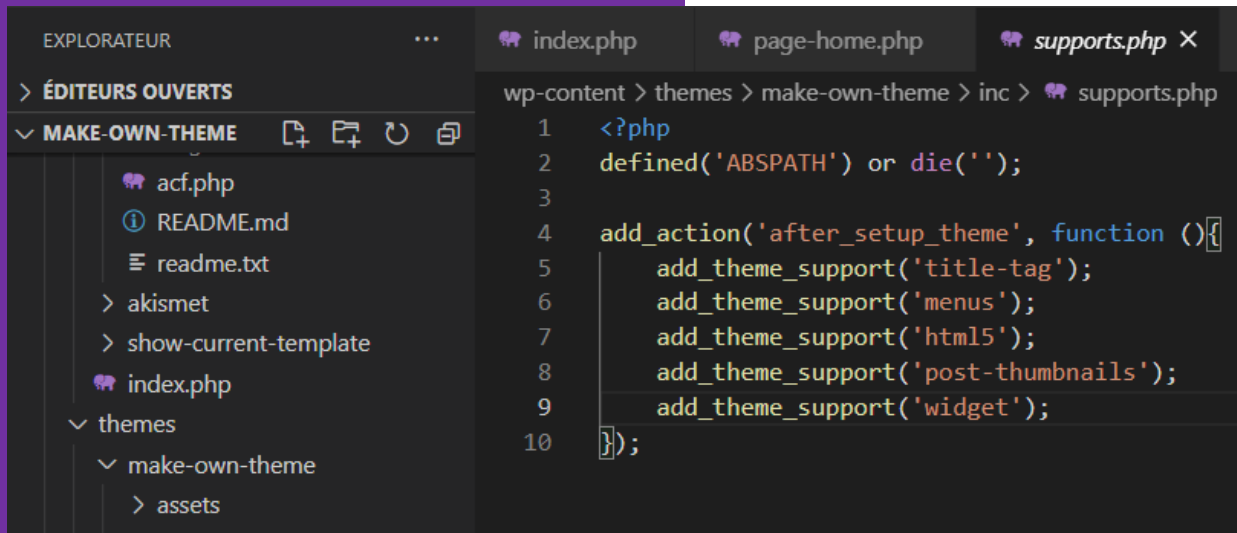
```
1  /*
2  * Custom Login Page CSS
3  */
4
5  body.login {
6      background-image: url('../images/background.jpg');
7      /* background-image: linear-gradient(60deg, #C835F4, #F
8      background-repeat: no-repeat;
9      background-attachment: fixed;
10     background-position: center;
11     background-size: cover;
12 }
13
14 /* Logo */
15 .login h1 a {
16     background: url('../images/mountain-logo-256.png') no-r
17     background-size: contain !important;
18 }
19
20 /* formulaire de connexion */
21 .login form {
22     background: ■ rgba(255, 255, 255, 0.86);
23     border-radius: 2px;
24 }
25
26 .login label {
27     font-size: 12px;
28     color: □ #555555;
29 }
```




Wordpress

Déclaration de la zone du menu de
navigation principal

Ajout du support pour le menu (supports.php)



The screenshot shows a code editor with a sidebar on the left and a main editor area on the right. The sidebar, titled 'EXPLORATEUR', shows a file tree for a project named 'MAKE-OWN-THEME'. The tree includes files like 'acf.php', 'README.md', 'readme.txt', and a 'themes' directory containing 'make-own-theme' and 'assets'. The main editor area shows the 'supports.php' file with the following PHP code:

```
1 <?php
2 defined('ABSPATH') or die('');
3
4 add_action('after_setup_theme', function () {
5     add_theme_support('title-tag');
6     add_theme_support('menus');
7     add_theme_support('html5');
8     add_theme_support('post-thumbnails');
9     add_theme_support('widget');
10 });
```

Pour pouvoir ajouter ou modifier les menus sur le backoffice, il faut ajouter le support dans le thème grâce à la fonction « `add_theme_support` » et en lui passant « `menus` » comme argument.

Ajout du support du menu (functions.php)

Une fois le support des menus ajouté dans « supports.php », il ne faut pas oublier le « require » dans le fichier « functions.php » pour être sûr que les ajouts seront faits.

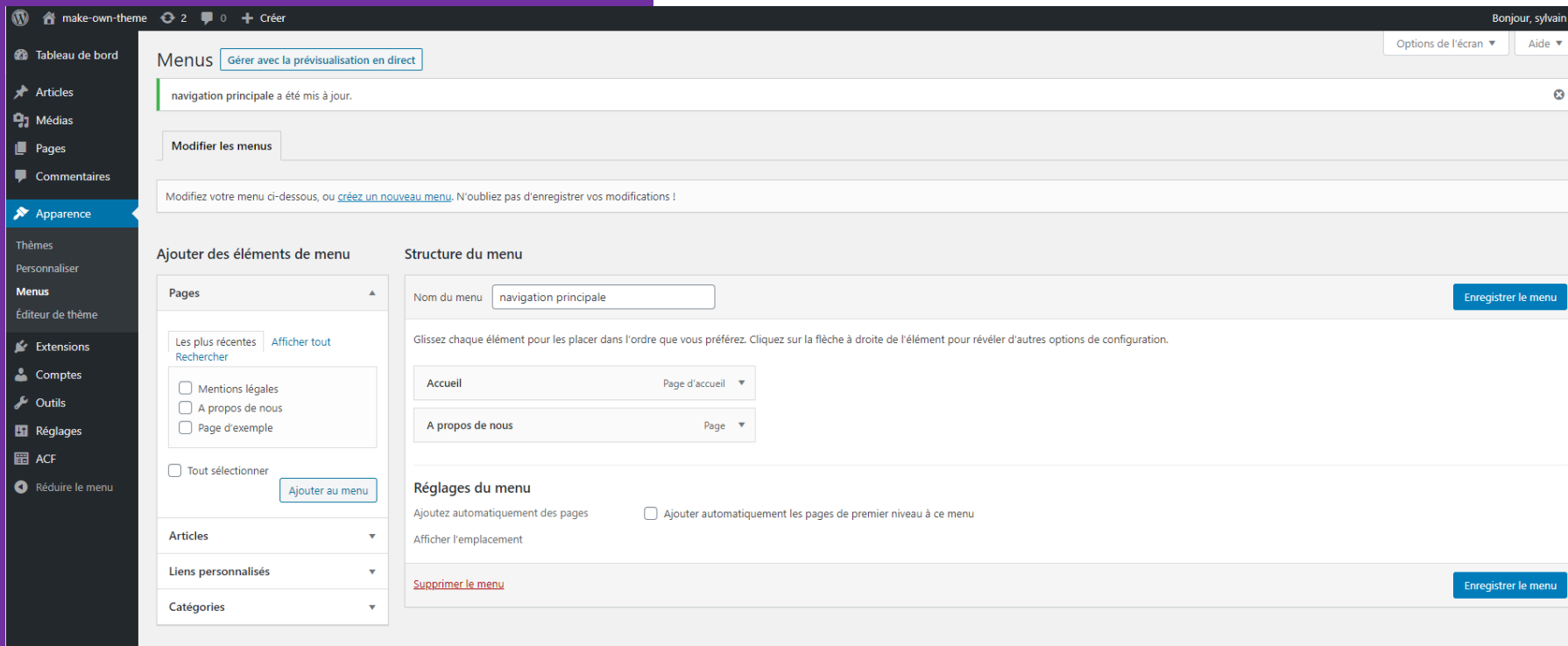
```
25  /*****  
26  * supports  
27  *****/  
28  require_once('inc/supports.php');
```

Création du menu dans le backoffice

Une fois que le support pour le menu est déclaré dans le thème, la zone de menu est à mettre en place en se rendant dans Apparence > Menus.

Il faut déterminer un nom pour le menu.

Puis ensuite, lui attribuer les pages correspondantes et enregistrer le menu.





Wordpress

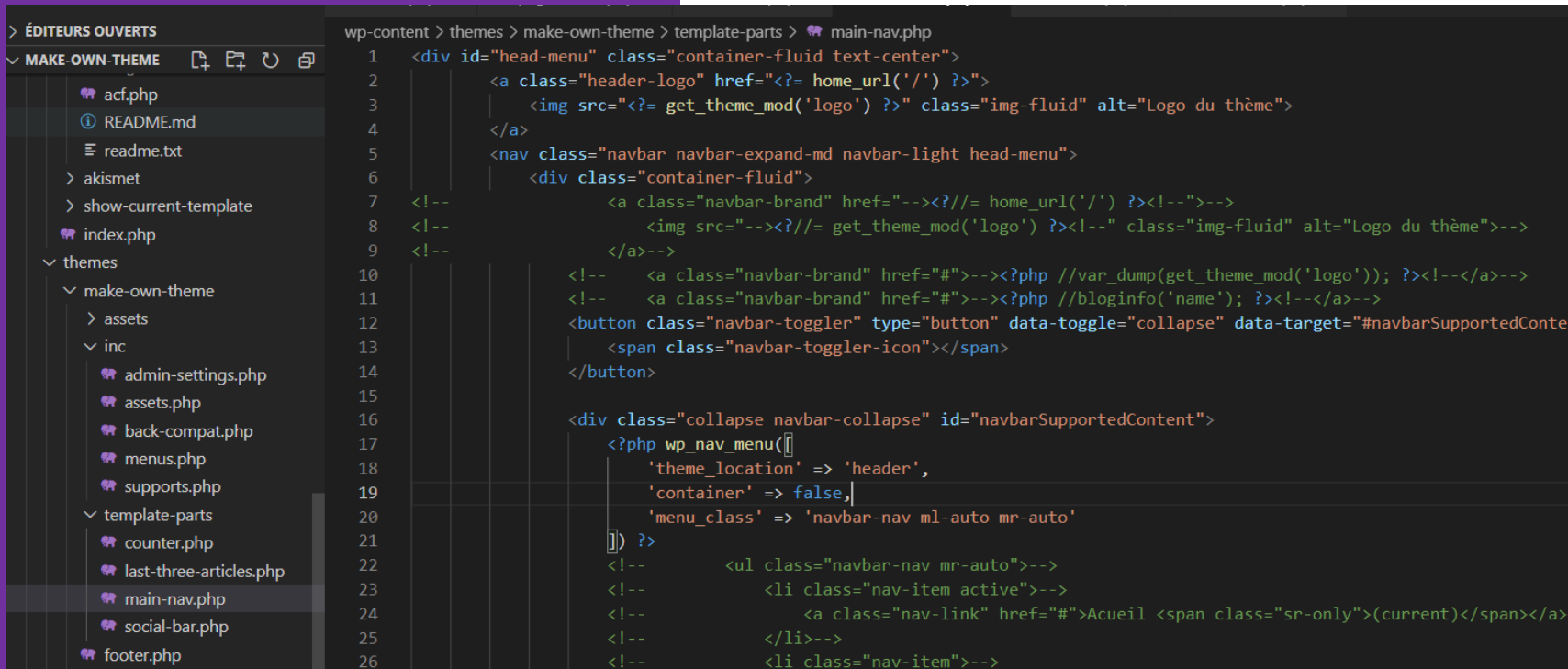
Intégration d'un menu simple

Mise en place du menu dans header.php

Pour le menu de navigation,
c'est au niveau du fichier
« header.php » qu'il faudra
ajouter avec la fonction
« get_template_part » l'appel au
fichier du template du menu.

```
26
27 <body id="page-top">
28 <header>
29     <?php
30         // Output the main navigation menu
31         get_template_part( 'template-parts/main-nav' );
32     ?>
33 </header>
```

Création du template pour le menu



The screenshot shows a code editor with two panels. The left panel displays the file structure of a WordPress theme, with the 'template-parts' directory expanded. The right panel shows the code for 'main-nav.php'.

```
wp-content > themes > make-own-theme > template-parts > main-nav.php
1 <div id="head-menu" class="container-fluid text-center">
2   <a class="header-logo" href="php home_url('/') ?">
3     
4   </a>
5   <nav class="navbar navbar-expand-md navbar-light head-menu">
6     <div class="container-fluid">
7       <!-- <a class="navbar-brand" href="--><?php home_url('/') ?><!-->-->
8       <!-- 
9       <!-- </a-->
10      <!-- <a class="navbar-brand" href="#"--><?php //var_dump(get_theme_mod('logo')); ?><!--</a-->
11      <!-- <a class="navbar-brand" href="#"--><?php //bloginfo('name'); ?><!--</a-->
12      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent">
13        <span class="navbar-toggler-icon"></span>
14      </button>
15
16      <div class="collapse navbar-collapse" id="navbarSupportedContent">
17        <?php wp_nav_menu([
18          'theme_location' => 'header',
19          'container' => false,
20          'menu_class' => 'navbar-nav ml-auto mr-auto'
21        ]) ?>
22        <!-- <ul class="navbar-nav mr-auto">-->
23        <!-- <li class="nav-item active">-->
24        <!-- <a class="nav-link" href="#">Accueil <span class="sr-only">(current)</span></a>-->
25        <!-- </li>-->
26        <!-- <li class="nav-item">-->
```

Le fichier « main-nav.php » est à créer dans le dossier « template-parts ».

Sur cette capture, c'est une barre de navigation de Bootstrap qui a été utilisée pour la structure.

La fonction « wp_nav_menu » permet d'appeler un menu en précisant la localisation du menu dans le thème et de rajouter des classes propres à Bootstrap pour styliser le menu.

Ajout d'un logo dynamique (functions.php)

Un petit plus qui peut être appréciable pour l'utilisateur, en lui permettant d'ajouter le logo qu'il souhaite dans le backoffice.

Pour ce faire, il faut commencer par faire un « require_once » du fichier « appearance.php » dans le fichier « functions.php ».

Ensuite, la création du fichier « appearance.php » se fait dans le dossier « inc » qui est à la racine du dossier du thème.

```
/* *****  
 * appearance (logo)  
 ***** */  
require_once('inc/appearance.php');
```

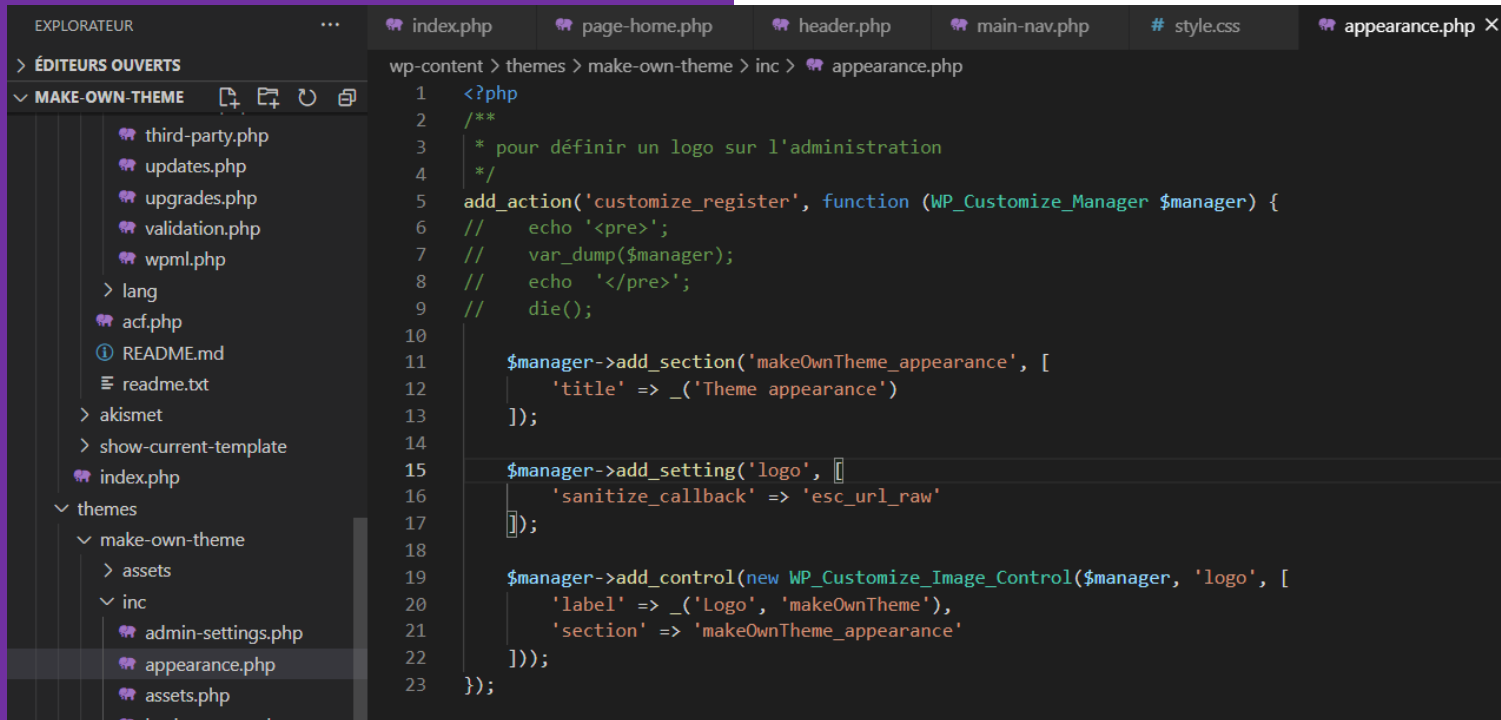
Ajout d'un logo dynamique (appearance.php)

Le fichier «appearance.php» va permettre d'ajouter une action qui donnera la possibilité d'ajouter un logo via le menu apparence dans le backoffice.

La première fonction « add_section » qui est branchée sur le manager va ajouter la section.

Ensuite, un paramètre va se greffer pour nettoyer l'url et s'assurer que c'est un slug.

Et pour finir, l'ajout de contrôles pour l'utilisateur du backoffice.



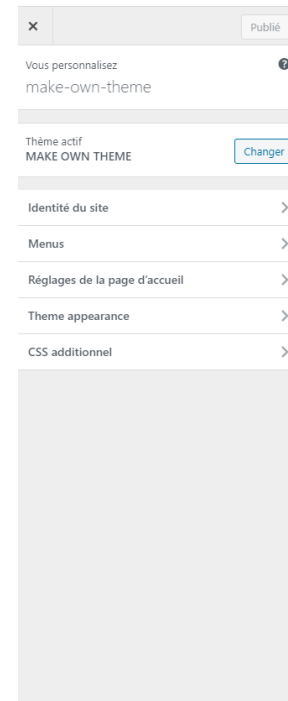
```
1 <?php
2 /**
3  * pour définir un logo sur l'administration
4  */
5 add_action('customize_register', function (WP_Customize_Manager $manager) {
6     // echo '<pre>';
7     // var_dump($manager);
8     // echo '</pre>';
9     // die();
10
11     $manager->add_section('makeOwnTheme_appearance', [
12         'title' => __('Theme appearance')
13     ]);
14
15     $manager->add_setting('logo', [
16         'sanitize_callback' => 'esc_url_raw'
17     ]);
18
19     $manager->add_control(new WP_Customize_Image_Control($manager, 'logo', [
20         'label' => __('Logo', 'makeOwnTheme'),
21         'section' => 'makeOwnTheme_appearance'
22     ]));
23 });
```

Ajout d'un logo dynamique (functions.php)

Pour ajouter un logo personnalisé, il faut aller dans « apparence » puis « personnaliser » dans le backoffice.

L'onglet « theme appearance » est désormais disponible. Il faut cliquer dessus pour pouvoir changer le logo.

Pour finir, un simple glisser/déposer de l'image qui servira de logo, un ajout du texte alternatif et un enregistrement des modifications permettront d'ajouter le logo sur les pages du site.

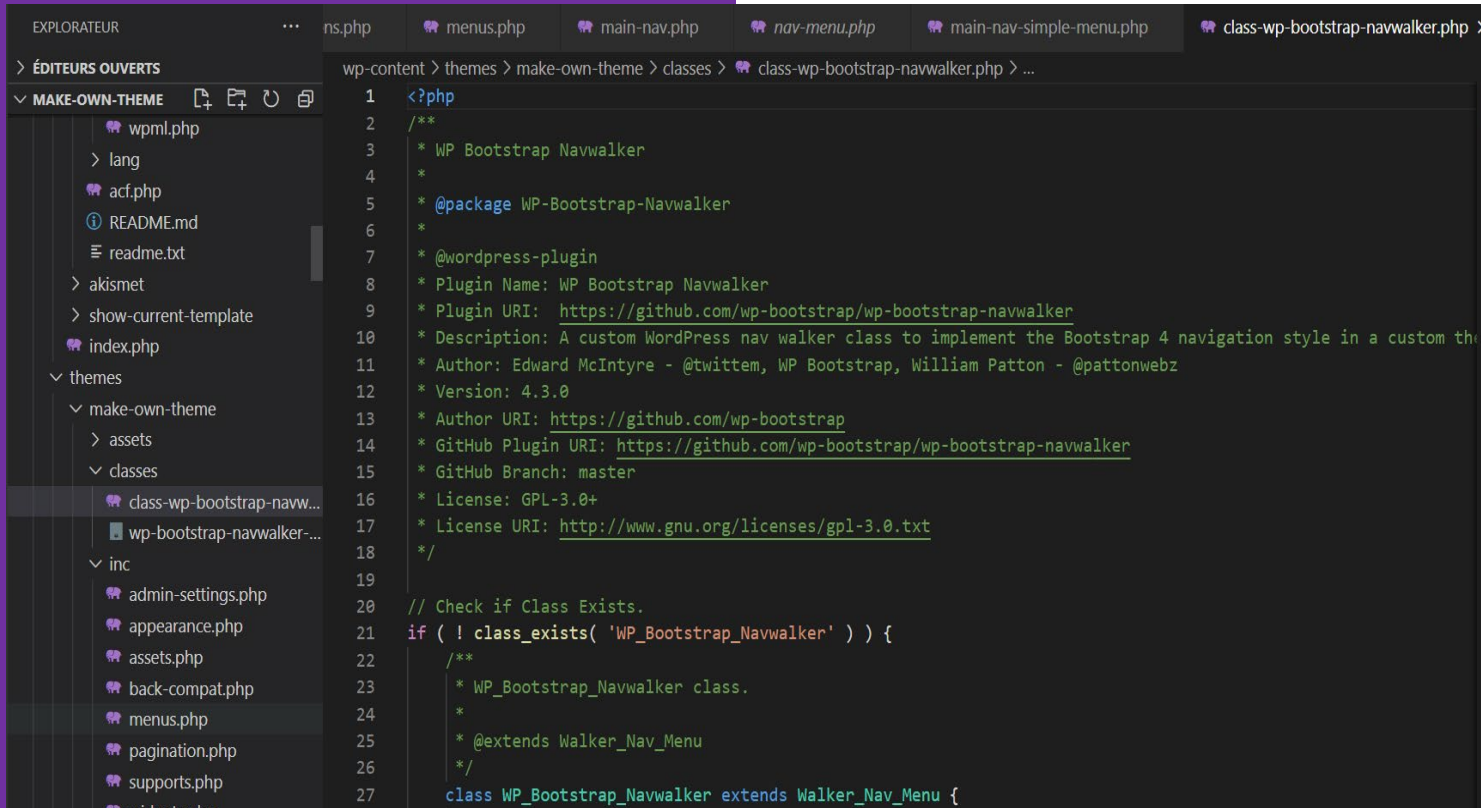




Wordpress

Intégration d'un menu complexe avec un walker pour Bootstrap 4

Télécharger le walker



The screenshot shows a code editor with a sidebar on the left displaying the file structure of a WordPress theme. The main editor area shows the content of the file `class-wp-bootstrap-navwalker.php`. The sidebar lists the following structure:

- wpml.php
- lang
- acf.php
- README.md
- readme.txt
- akismet
- show-current-template
- index.php
- themes
 - make-own-theme
 - assets
 - classes
 - class-wp-bootstrap-navwalker.php
 - wp-bootstrap-navwalker...
 - inc
 - admin-settings.php
 - appearance.php
 - assets.php
 - back-compat.php
 - menus.php
 - pagination.php
 - supports.php
 - widgets.php

The main editor area shows the following code:

```
1 <?php
2 /**
3  * WP Bootstrap Navwalker
4  *
5  * @package WP-Bootstrap-Navwalker
6  *
7  * @wordpress-plugin
8  * Plugin Name: WP Bootstrap Navwalker
9  * Plugin URI: https://github.com/wp-bootstrap/wp-bootstrap-navwalker
10 * Description: A custom WordPress nav walker class to implement the Bootstrap 4 navigation style in a custom theme
11 * Author: Edward McIntyre - @twittem, WP Bootstrap, William Patton - @pattonwebz
12 * Version: 4.3.0
13 * Author URI: https://github.com/wp-bootstrap
14 * GitHub Plugin URI: https://github.com/wp-bootstrap/wp-bootstrap-navwalker
15 * GitHub Branch: master
16 * License: GPL-3.0+
17 * License URI: http://www.gnu.org/licenses/gpl-3.0.txt
18 */
19
20 // Check if Class Exists.
21 if ( ! class_exists( 'WP_Bootstrap_Navwalker' ) ) {
22     /**
23      * WP_Bootstrap_Navwalker class.
24      *
25      * @extends Walker_Nav_Menu
26      */
27     class WP_Bootstrap_Navwalker extends Walker_Nav_Menu {
```

Pour télécharger le « walker » via ce lien <https://github.com/wp-bootstrap/wp-bootstrap-navwalker> pour avoir la possibilité de customisé encore plus un menu dans Wordpress par exemple en rajoutant de la profondeur dans la hiérarchie des liens du menu.

Une fois le walker téléchargé, il faut dézipper le dossier pour récupérer le fichier qui est nécessaire.

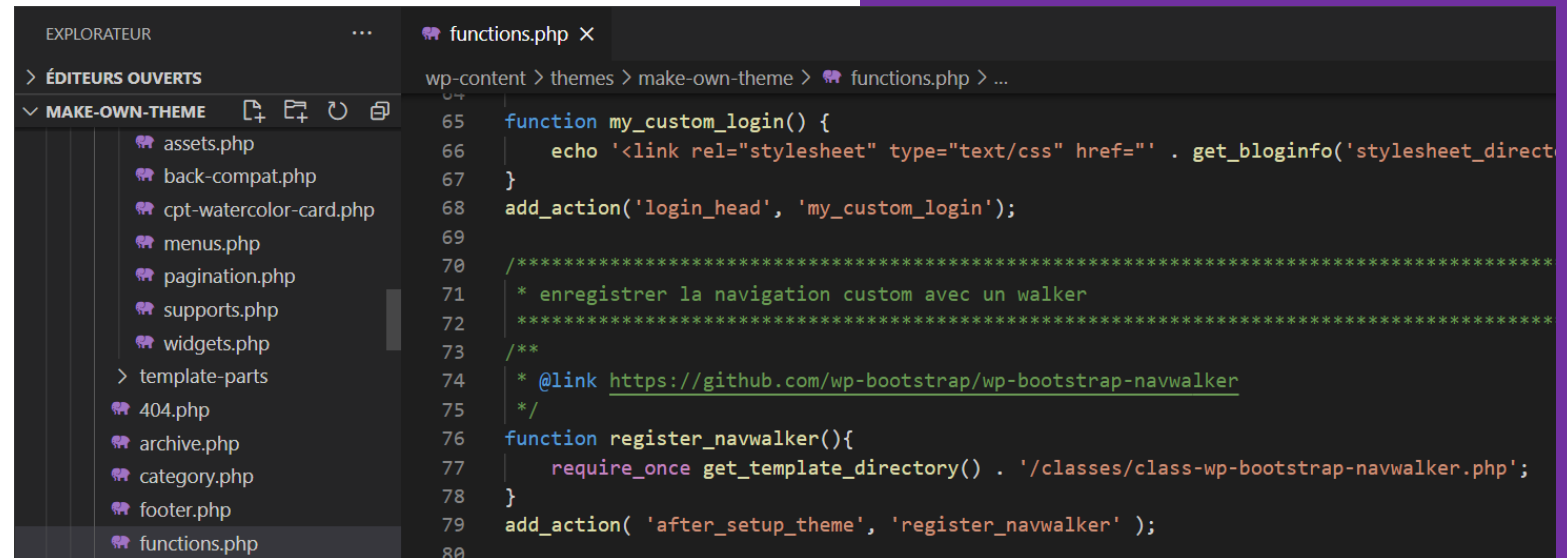
Ici, c'est le fichier: « class-wp-bootstrap-navwalker.php ».

Insertion dans “functions.php”

Il faut désormais mettre le fichier téléchargé dans le dossier « classes ».

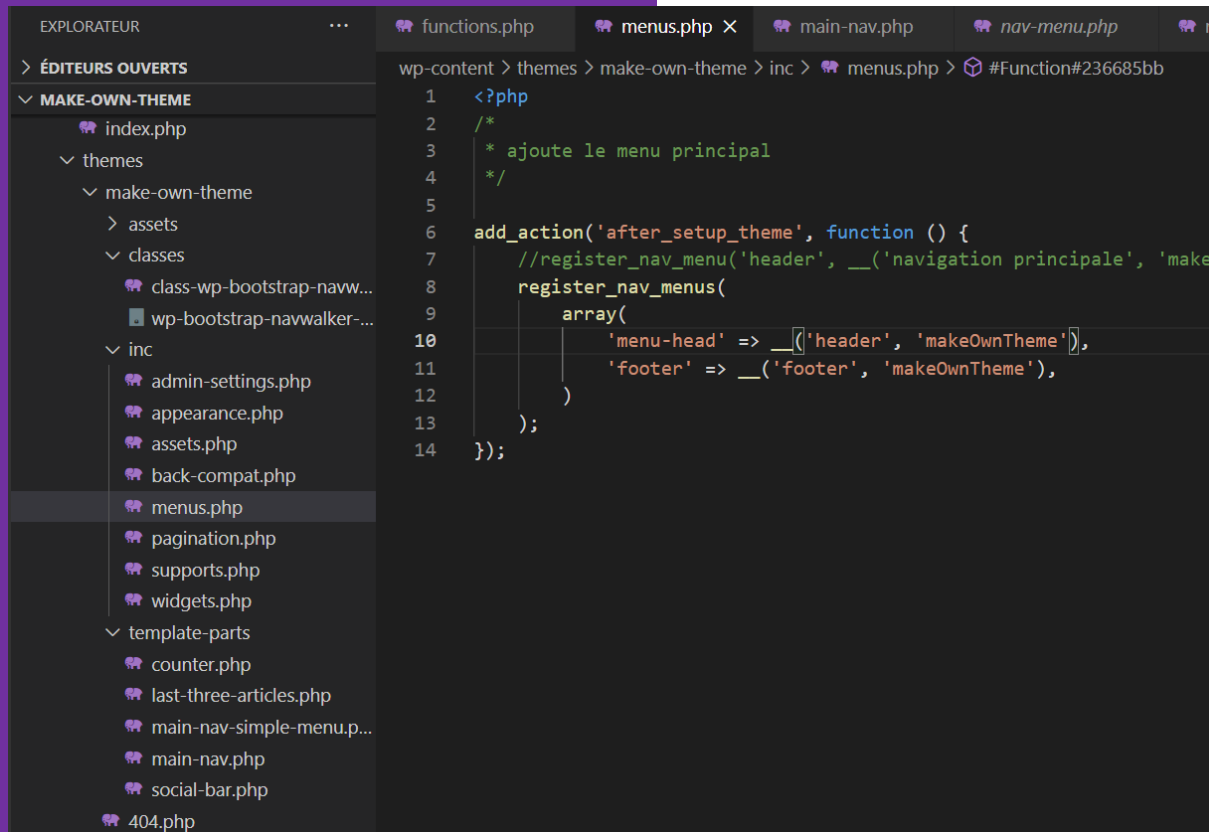
Il faut enregistrer ce « walker » dans le « hook » de Wordpress pour y avoir accès.

Penser à bien récupérer le bon fichier grâce à la fonction « `get_template_directory` » puis l'url qui correspond à la structure ou se trouve le fichier.



```
65 function my_custom_login() {
66     echo '<link rel="stylesheet" type="text/css" href="' . get_bloginfo('stylesheet_direct
67 }
68 add_action('login_head', 'my_custom_login');
69
70 /*****
71  * enregistrer la navigation custom avec un walker
72  *****/
73 /**
74  * @link https://github.com/wp-bootstrap/wp-bootstrap-navwalker
75  */
76 function register_navwalker(){
77     require_once get_template_directory() . '/classes/class-wp-bootstrap-navwalker.php';
78 }
79 add_action( 'after_setup_theme', 'register_navwalker' );
80
```

Enregistrement des menus



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure for a WordPress theme, with 'menus.php' selected. The code editor shows the following PHP code in 'functions.php':

```
1 <?php
2 /*
3  * ajoute le menu principal
4  */
5
6 add_action('after_setup_theme', function () {
7     //register_nav_menu('header', __('navigation principale', 'makeO
8     register_nav_menus(
9         array(
10            'menu-head' => __(['header', 'makeOwnTheme']),
11            'footer' => __('footer', 'makeOwnTheme'),
12        )
13    );
14 });
```

Comme pour le menu simple, il faut veiller à bien enregistrer le menu pour Wordpress.

Avec la fonction « register_nav_menus » qui permet d'enregistrer plusieurs menus.

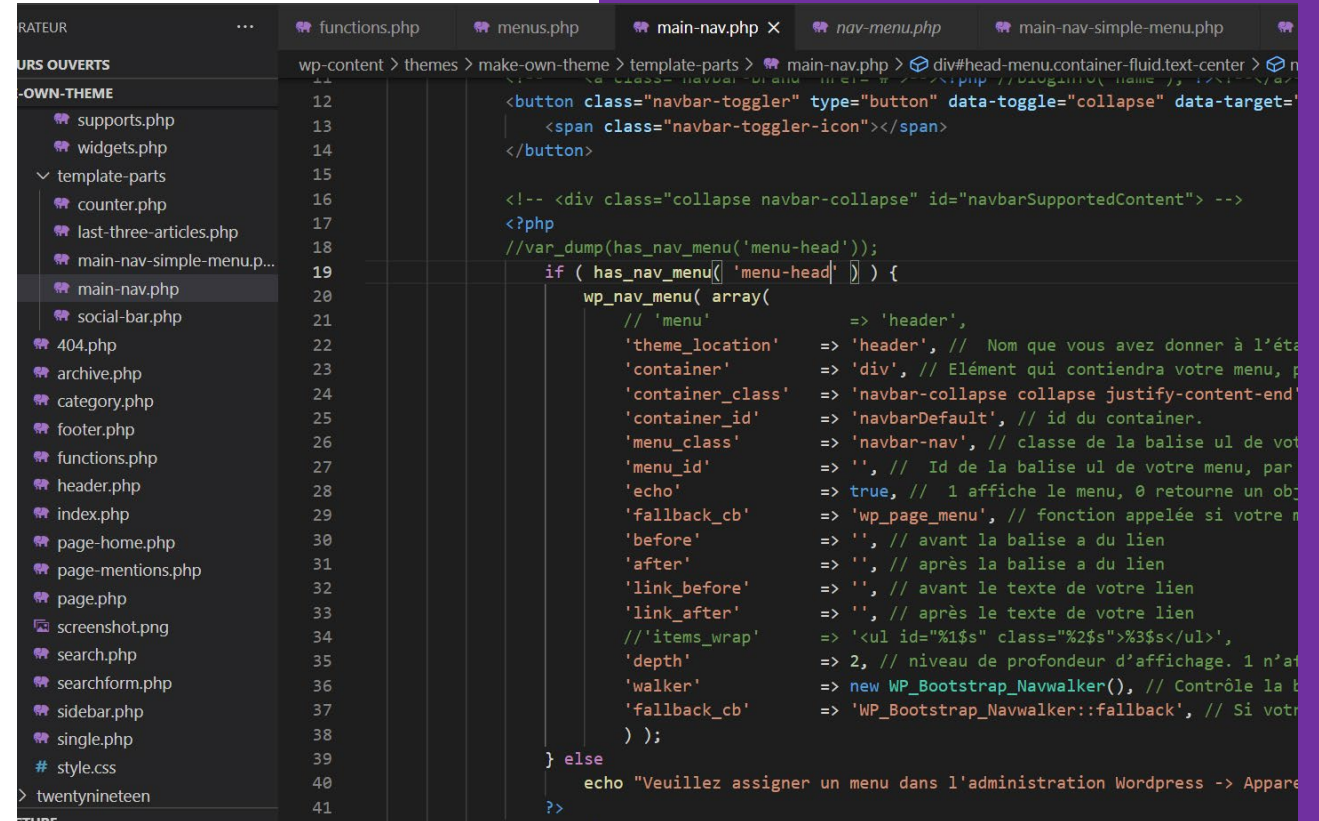
Cette fonction prends en paramètres un tableau PHP qui comprend a chaque ligne le nom du menu et sa localisation dans le thème.

Mise en place du menu dans le header

Tous d'abord, la vérification de l'existence du menu avec la fonction « `has_nav_menu` ».

Ensuite avec la fonction « `wp_nav_menu` », un tableau PHP permet de configurer les différentes options du menu.

Comme sa localisation, les classes utilisées dans le menu et le « walker » utilisé pour contrôler la barre de navigation.



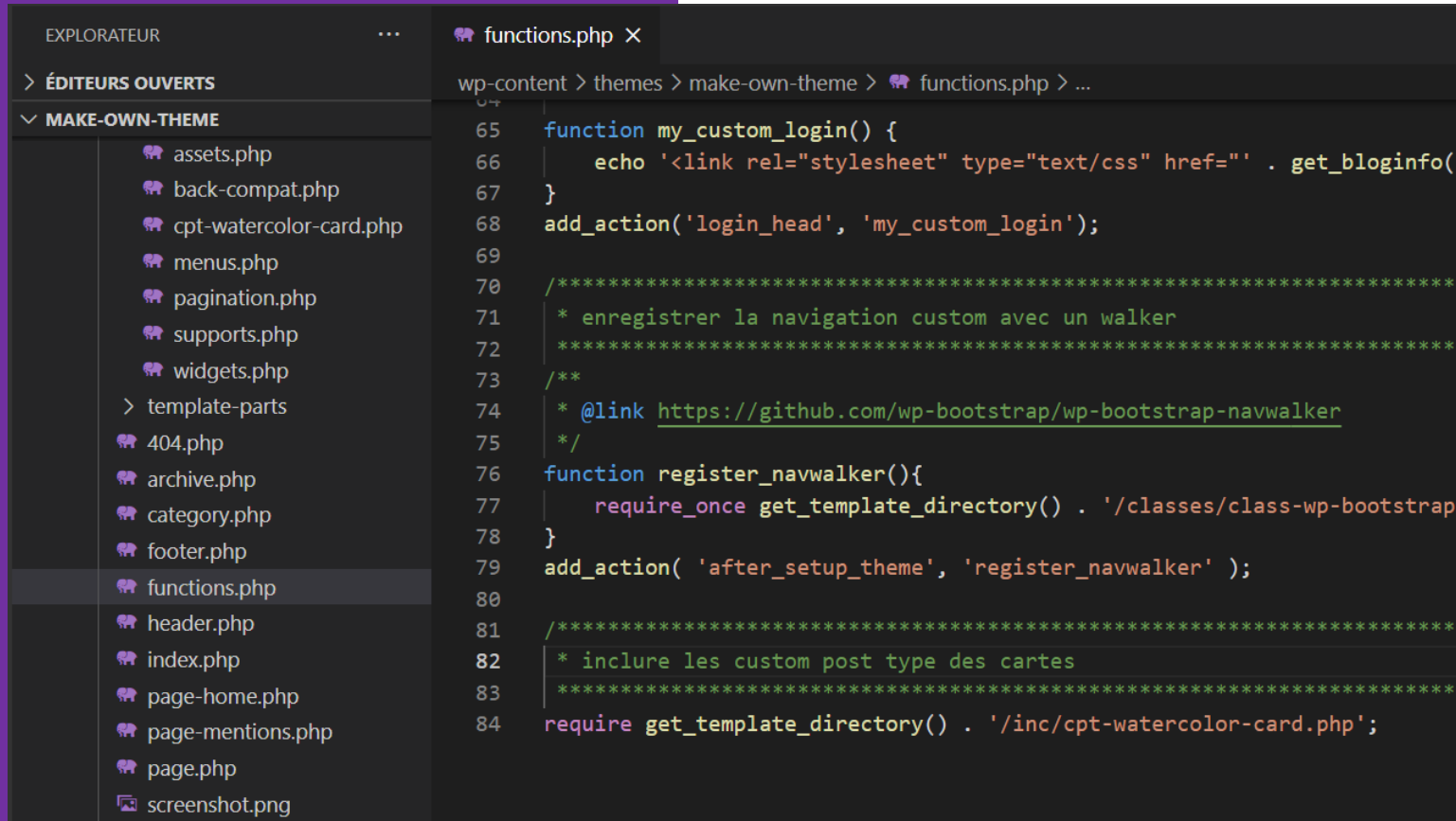
```
wp-content > themes > make-own-theme > template-parts > main-nav.php > div#head-menu.container-fluid.text-center > n
12 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#
13 <span class="navbar-toggler-icon"></span>
14 </button>
15
16 <!-- <div class="collapse navbar-collapse" id="navbarSupportedContent"> -->
17 <?php
18 //var_dump(has_nav_menu('menu-head'));
19 if ( has_nav_menu('menu-head') ) {
20     wp_nav_menu( array(
21         // 'menu' => 'header',
22         'theme_location' => 'header', // Nom que vous avez donné à l'éta
23         'container' => 'div', // Élément qui contiendra votre menu, p
24         'container_class' => 'navbar-collapse collapse justify-content-end'
25         'container_id' => 'navbarDefault', // id du container.
26         'menu_class' => 'navbar-nav', // classe de la balise ul de vot
27         'menu_id' => '', // Id de la balise ul de votre menu, par
28         'echo' => true, // 1 affiche le menu, 0 retourne un obj
29         'fallback_cb' => 'wp_page_menu', // fonction appelée si votre m
30         'before' => '', // avant la balise a du lien
31         'after' => '', // après la balise a du lien
32         'link_before' => '', // avant le texte de votre lien
33         'link_after' => '', // après le texte de votre lien
34         // 'items_wrap' => '<ul id="%1$s" class="%2$s">%3$s</ul>',
35         'depth' => 2, // niveau de profondeur d'affichage. 1 n'at
36         'walker' => new WP_Bootstrap_Navwalker(), // Contrôle la b
37         'fallback_cb' => 'WP_Bootstrap_Navwalker::fallback', // Si votr
38     ) );
39 } else
40     echo "Veuillez assigner un menu dans l'administration Wordpress -> Appare
41 >
```



Wordpress

Création from Scratch d'un Custom Post Type (functions.php) avec ses taxonomies

Inclure les “custom post type”



```
65 function my_custom_login() {
66     echo '<link rel="stylesheet" type="text/css" href="' . get_bloginfo(
67 }
68 add_action('login_head', 'my_custom_login');
69
70 /*****
71  * enregistrer la navigation custom avec un walker
72  *****/
73 /**
74  * @link https://github.com/wp-bootstrap/wp-bootstrap-navwalker
75  */
76 function register_navwalker(){
77     require_once get_template_directory() . '/classes/class-wp-bootstrap-
78 }
79 add_action( 'after_setup_theme', 'register_navwalker' );
80
81 /*****
82  * inclure les custom post type des cartes
83  *****/
84 require get_template_directory() . '/inc/cpt-watercolor-card.php';
```

Qu'est-ce qu'un « custom post type » ?

C'est un type de contenu customisé qui adopte le comportement des articles de Wordpress.

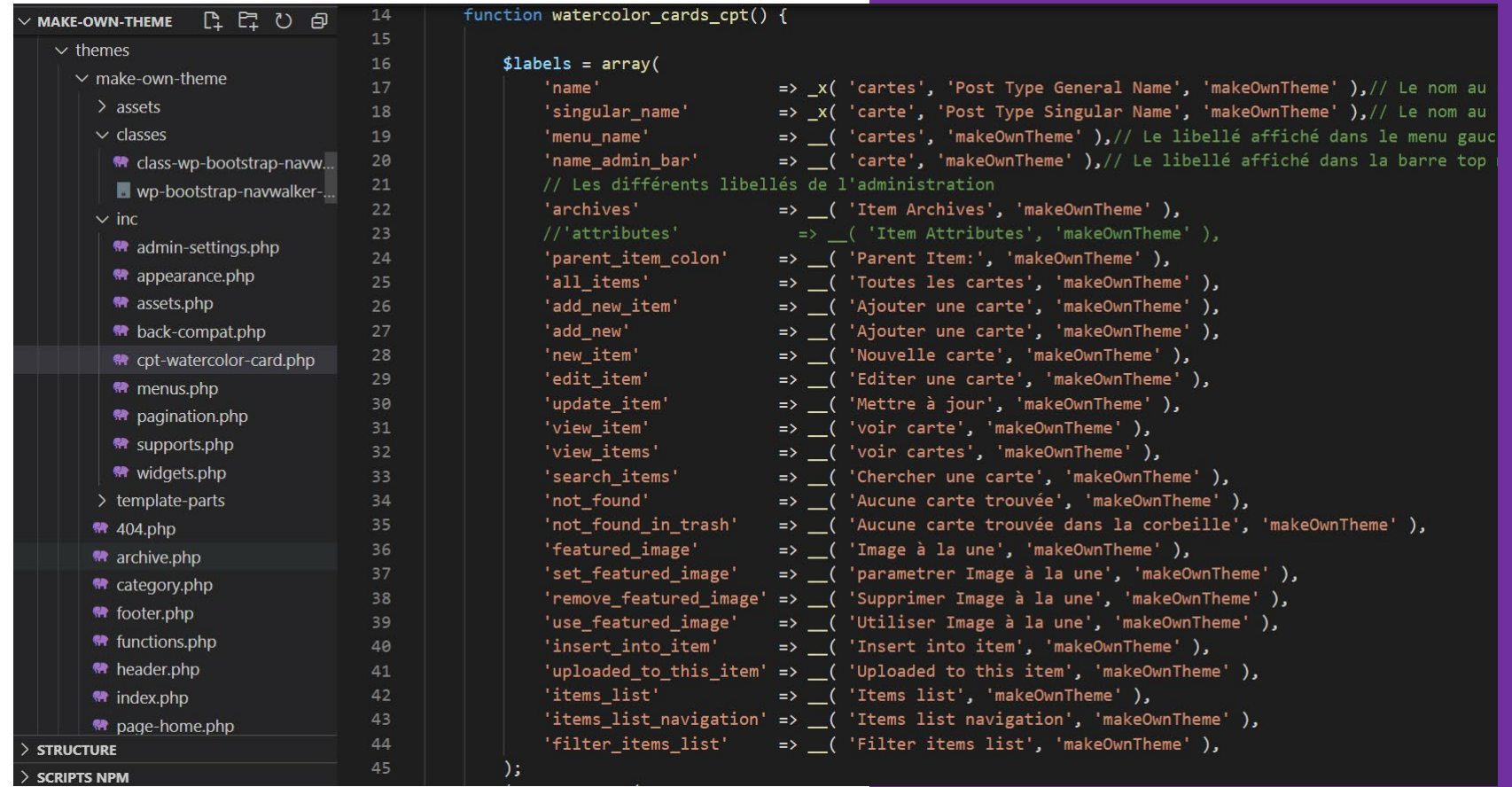
Pour commencer, il faut créer le fichier « cpt-watercolor-card » dans le dossier « inc » puis ne pas oublier de l'inclure dans le fichier « functions.php ».

Le fichier “cpt-watercolor-card”

Ici, le « custom post type » est nommé « carte » mais il est bien sur possible de créer un tout autre en changeant le nom et les paramètres.

Le tableau « labels » contient tous les paramètres à mettre en place pour gérer le nom au singulier et au pluriel du « custom post type » .

Mais aussi tous les affichages des textes dans le backoffice.



```
function watercolor_cards_cpt() {  
    $labels = array(  
        'name' => _x( 'cartes', 'Post Type General Name', 'makeOwnTheme' ), // Le nom au  
        'singular_name' => _x( 'carte', 'Post Type Singular Name', 'makeOwnTheme' ), // Le nom au  
        'menu_name' => __( 'cartes', 'makeOwnTheme' ), // Le libellé affiché dans le menu gauc  
        'name_admin_bar' => __( 'carte', 'makeOwnTheme' ), // Le libellé affiché dans la barre top  
        // Les différents libellés de l'administration  
        'archives' => __( 'Item Archives', 'makeOwnTheme' ),  
        // 'attributes' => __( 'Item Attributes', 'makeOwnTheme' ),  
        'parent_item_colon' => __( 'Parent Item:', 'makeOwnTheme' ),  
        'all_items' => __( 'Toutes les cartes', 'makeOwnTheme' ),  
        'add_new_item' => __( 'Ajouter une carte', 'makeOwnTheme' ),  
        'add_new' => __( 'Ajouter une carte', 'makeOwnTheme' ),  
        'new_item' => __( 'Nouvelle carte', 'makeOwnTheme' ),  
        'edit_item' => __( 'Editer une carte', 'makeOwnTheme' ),  
        'update_item' => __( 'Mettre à jour', 'makeOwnTheme' ),  
        'view_item' => __( 'voir carte', 'makeOwnTheme' ),  
        'view_items' => __( 'voir cartes', 'makeOwnTheme' ),  
        'search_items' => __( 'Chercher une carte', 'makeOwnTheme' ),  
        'not_found' => __( 'Aucune carte trouvée', 'makeOwnTheme' ),  
        'not_found_in_trash' => __( 'Aucune carte trouvée dans la corbeille', 'makeOwnTheme' ),  
        'featured_image' => __( 'Image à la une', 'makeOwnTheme' ),  
        'set_featured_image' => __( 'parametrer Image à la une', 'makeOwnTheme' ),  
        'remove_featured_image' => __( 'Supprimer Image à la une', 'makeOwnTheme' ),  
        'use_featured_image' => __( 'Utiliser Image à la une', 'makeOwnTheme' ),  
        'insert_into_item' => __( 'Insert into item', 'makeOwnTheme' ),  
        'uploaded_to_this_item' => __( 'Uploaded to this item', 'makeOwnTheme' ),  
        'items_list' => __( 'Items list', 'makeOwnTheme' ),  
        'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),  
        'filter_items_list' => __( 'Filter items list', 'makeOwnTheme' ),  
    );  
}
```

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure for a WordPress theme named 'MAKE-OWN-THEME'. The 'themes' directory contains a 'make-own-theme' subdirectory, which includes 'assets', 'classes', and 'inc' subdirectories. The 'inc' directory contains several PHP files, including 'cpt-watercolor-card.php', which is highlighted. The code editor on the right shows the function 'watercolor_cards_cpt()' which defines an array of labels for a custom post type named 'cartes'. The labels include 'name', 'singular_name', 'menu_name', 'name_admin_bar', 'archives', 'parent_item_colon', 'all_items', 'add_new_item', 'add_new', 'new_item', 'edit_item', 'update_item', 'view_item', 'view_items', 'search_items', 'not_found', 'not_found_in_trash', 'featured_image', 'set_featured_image', 'remove_featured_image', 'use_featured_image', 'insert_into_item', 'uploaded_to_this_item', 'items_list', 'items_list_navigation', and 'filter_items_list'. Each label is assigned a value using the WordPress localization functions.

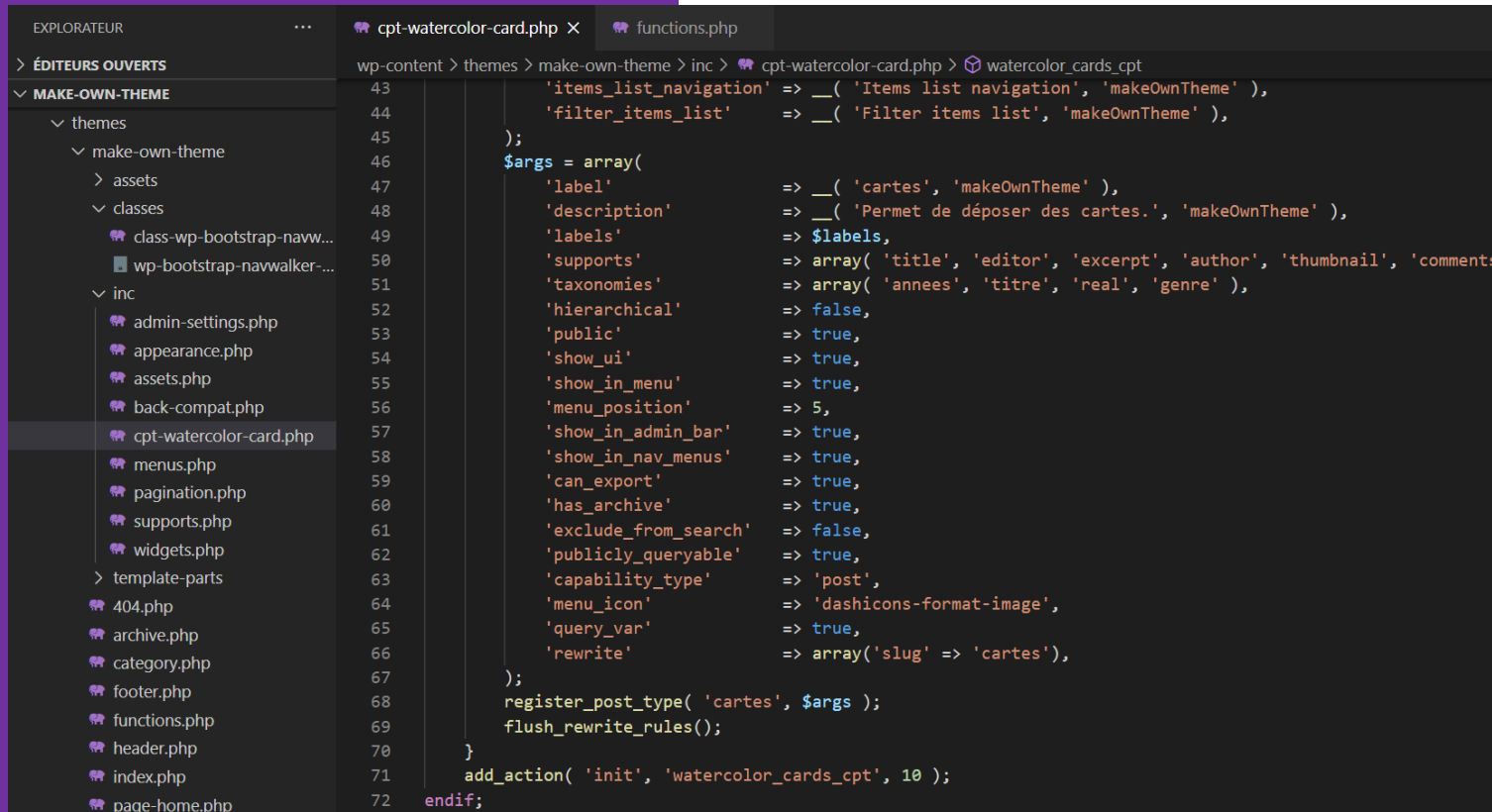
Les arguments du custom post type

Après avoir mis en place le tableau « labels », il faut créer le tableau d'arguments pour le « custom post type ».

Ici, il faut déterminer le label, la description, les supports autorisés (titre, éditeur, image à la une) et les taxonomies utilisées.

Le paramètre « menu_position » permet de définir une position dans le menu du backoffice.

Il ne reste plus qu'à l'enregistrer grâce à la fonction « register_post_type ».



```
43 'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),
44 'filter_items_list'    => __( 'Filter items list', 'makeOwnTheme' ),
45 );
46 $args = array(
47     'label'              => __( 'cartes', 'makeOwnTheme' ),
48     'description'       => __( 'Permet de déposer des cartes.', 'makeOwnTheme' ),
49     'labels'             => $labels,
50     'supports'           => array( 'title', 'editor', 'excerpt', 'author', 'thumbnail', 'comments' ),
51     'taxonomies'         => array( 'annees', 'titre', 'real', 'genre' ),
52     'hierarchical'       => false,
53     'public'             => true,
54     'show_ui'            => true,
55     'show_in_menu'       => true,
56     'menu_position'      => 5,
57     'show_in_admin_bar'  => true,
58     'show_in_nav_menus'  => true,
59     'can_export'         => true,
60     'has_archive'        => true,
61     'exclude_from_search'=> false,
62     'publicly_queryable' => true,
63     'capability_type'    => 'post',
64     'menu_icon'          => 'dashicons-format-image',
65     'query_var'          => true,
66     'rewrite'            => array( 'slug' => 'cartes' ),
67 );
68 register_post_type( 'cartes', $args );
69 flush_rewrite_rules();
70 }
71 add_action( 'init', 'watercolor_cards_cpt', 10 );
72 endif;
```

Taxonomie “année”

Comme pour le « custom post type », ses taxonomies doivent être enregistrées.

Il faut déclarer deux tableaux, un pour les libellés et un pour les arguments.

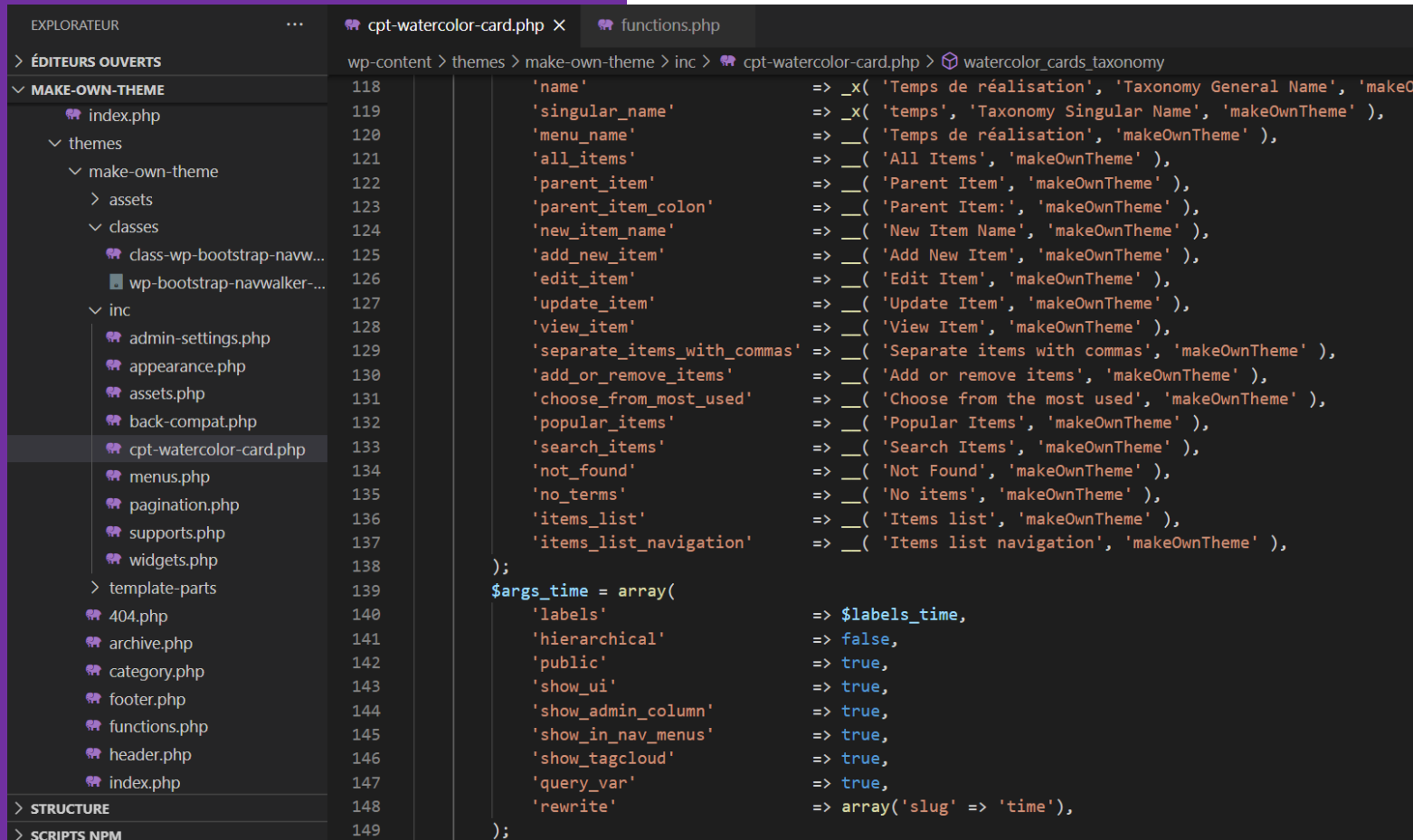
Pour la taxonomie « année », il faut déterminer les affichages pour les textes, sa hiérarchie, si son affichage est publique ou non, etc...

```
EXPLOREUR
...
cpt-watercolor-card.php X functions.php
> Éditeurs ouverts
v MAKE-OWN-THEME
  index.php
  themes
    make-own-theme
      assets
      classes
        class-wp-bootstrap-navw...
        wp-bootstrap-navwalker-...
      inc
        admin-settings.php
        appearance.php
        assets.php
        back-compat.php
        cpt-watercolor-card.php
        menus.php
        pagination.php
        supports.php
        widgets.php
      template-parts
        404.php
        archive.php
        category.php
        footer.php
        functions.php
        header.php
        index.php
  > STRUCTURE
  > SCRIPTS NPM

wp-content > themes > make-own-theme > inc > cpt-watercolor-card.php > watercolor_cards_taxonomy

81 $labels_annee = array(
82     'name' => __( 'Années', 'Taxonomy General Name', 'makeOwnTheme' ),
83     'singular_name' => __( 'Année', 'Taxonomy Singular Name', 'makeOwnTheme' ),
84     'menu_name' => __( 'Années', 'makeOwnTheme' ),
85     'all_items' => __( 'All Items', 'makeOwnTheme' ),
86     'parent_item' => __( 'Parent Item', 'makeOwnTheme' ),
87     'parent_item_colon' => __( 'Parent Item:', 'makeOwnTheme' ),
88     'new_item_name' => __( 'New Item Name', 'makeOwnTheme' ),
89     'add_new_item' => __( 'Add New Item', 'makeOwnTheme' ),
90     'edit_item' => __( 'Edit Item', 'makeOwnTheme' ),
91     'update_item' => __( 'Update Item', 'makeOwnTheme' ),
92     'view_item' => __( 'View Item', 'makeOwnTheme' ),
93     'separate_items_with_commas' => __( 'Separate items with commas', 'makeOwnTheme' ),
94     'add_or_remove_items' => __( 'Add or remove items', 'makeOwnTheme' ),
95     'choose_from_most_used' => __( 'Choose from the most used', 'makeOwnTheme' ),
96     'popular_items' => __( 'Popular Items', 'makeOwnTheme' ),
97     'search_items' => __( 'Search Items', 'makeOwnTheme' ),
98     'not_found' => __( 'Not Found', 'makeOwnTheme' ),
99     'no_terms' => __( 'No items', 'makeOwnTheme' ),
100     'items_list' => __( 'Items list', 'makeOwnTheme' ),
101     'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),
102 );
103
104 $args_annee = array(
105     'labels' => $labels_annee,
106     'hierarchical' => false, // Si 'hierarchical' est défini à false, notre taxonom
107     'public' => true,
108     'show_ui' => true,
109     'show_admin_column' => true,
110     'show_in_nav_menus' => true,
111     'show_tagcloud' => true,
112     'query_var' => true,
113     'rewrite' => array('slug' => 'annees'),
114 );
115
116 register_taxonomy( 'watercolor_cards_taxonomy', 'cpt-watercolor-card.php', $args_annee );
```

Taxonomie “temps”



```
EXPLOREUR
... cpt-watercolor-card.php X functions.php
> ÉDITEURS OUVERTS
  MAKE-OWN-THEME
    index.php
    themes
      make-own-theme
        assets
        classes
          class-wp-bootstrap-navw...
          wp-bootstrap-navwalker-...
        inc
          admin-settings.php
          appearance.php
          assets.php
          back-compat.php
          cpt-watercolor-card.php
          menus.php
          pagination.php
          supports.php
          widgets.php
        template-parts
          404.php
          archive.php
          category.php
          footer.php
          functions.php
          header.php
          index.php
  STRUCTURE
  SCRIPTS NPM

wp-content > themes > make-own-theme > inc > cpt-watercolor-card.php > watercolor_cards_taxonomy
118 'name' => __( 'Temps de réalisation', 'Taxonomy General Name', 'makeOw
119 'singular_name' => __( 'temps', 'Taxonomy Singular Name', 'makeOwnTheme' ),
120 'menu_name' => __( 'Temps de réalisation', 'makeOwnTheme' ),
121 'all_items' => __( 'All Items', 'makeOwnTheme' ),
122 'parent_item' => __( 'Parent Item', 'makeOwnTheme' ),
123 'parent_item_colon' => __( 'Parent Item:', 'makeOwnTheme' ),
124 'new_item_name' => __( 'New Item Name', 'makeOwnTheme' ),
125 'add_new_item' => __( 'Add New Item', 'makeOwnTheme' ),
126 'edit_item' => __( 'Edit Item', 'makeOwnTheme' ),
127 'update_item' => __( 'Update Item', 'makeOwnTheme' ),
128 'view_item' => __( 'View Item', 'makeOwnTheme' ),
129 'separate_items_with_commas' => __( 'Separate items with commas', 'makeOwnTheme' ),
130 'add_or_remove_items' => __( 'Add or remove items', 'makeOwnTheme' ),
131 'choose_from_most_used' => __( 'Choose from the most used', 'makeOwnTheme' ),
132 'popular_items' => __( 'Popular Items', 'makeOwnTheme' ),
133 'search_items' => __( 'Search Items', 'makeOwnTheme' ),
134 'not_found' => __( 'Not Found', 'makeOwnTheme' ),
135 'no_terms' => __( 'No items', 'makeOwnTheme' ),
136 'items_list' => __( 'Items list', 'makeOwnTheme' ),
137 'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),
138 );
139 $args_time = array(
140     'labels' => $labels_time,
141     'hierarchical' => false,
142     'public' => true,
143     'show_ui' => true,
144     'show_admin_column' => true,
145     'show_in_nav_menus' => true,
146     'show_tagcloud' => true,
147     'query_var' => true,
148     'rewrite' => array('slug' => 'time'),
149 );
```

Une fois la taxonomie « annee » déclarée, c’est au tour de la taxonomie « temps ».

Il faut déclarer deux tableaux, un pour les libellés et un pour les arguments.

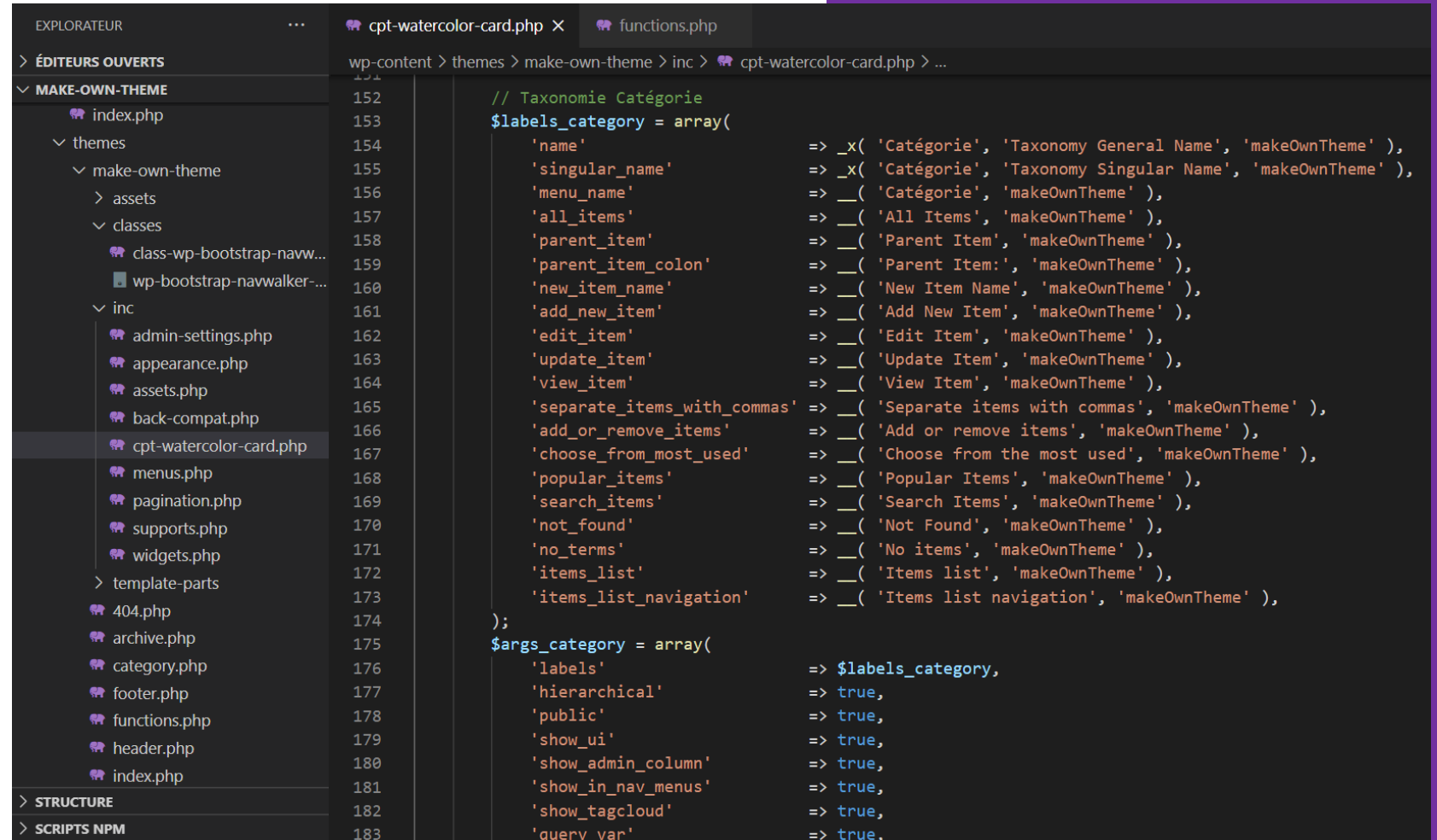
Pour la taxonomie « temps », il faut déterminer les affichages pour les textes, sa hiérarchie, si son affichage est publique ou non, etc...

Taxonomie «catégorie»

Une fois la taxonomie « temps » déclarée, c'est au tour de la taxonomie « catégorie ».

Il faut déclarer deux tableaux, un pour les libellés et un pour les arguments.

Pour la taxonomie « catégorie », il faut déterminer les affichages pour les textes, sa hiérarchie, si son affichage est publique ou non, etc...

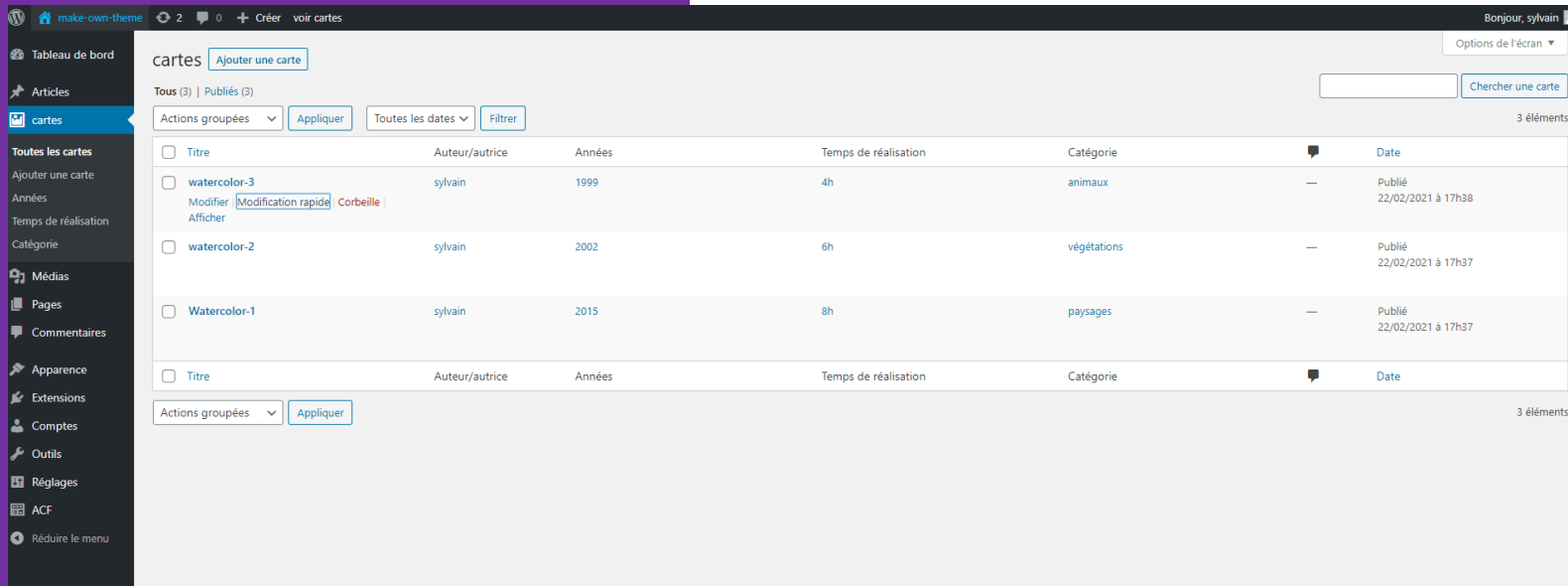


```
EXPLOREUR  ...  cpt-watercolor-card.php X  functions.php
> ÉDITEURS OUVERTS
  MAKE-OWN-THEME
    index.php
    themes
      make-own-theme
        assets
        classes
          class-wp-bootstrap-navw...
          wp-bootstrap-navwalker-...
        inc
          admin-settings.php
          appearance.php
          assets.php
          back-compat.php
          cpt-watercolor-card.php
          menus.php
          pagination.php
          supports.php
          widgets.php
        template-parts
          404.php
          archive.php
          category.php
          footer.php
          functions.php
          header.php
          index.php
  STRUCTURE
  SCRIPTS NPM

wp-content > themes > make-own-theme > inc > cpt-watercolor-card.php > ...

152 // Taxonomie Catégorie
153 $labels_category = array(
154     'name' => _x( 'Catégorie', 'Taxonomy General Name', 'makeOwnTheme' ),
155     'singular_name' => _x( 'Catégorie', 'Taxonomy Singular Name', 'makeOwnTheme' ),
156     'menu_name' => __( 'Catégorie', 'makeOwnTheme' ),
157     'all_items' => __( 'All Items', 'makeOwnTheme' ),
158     'parent_item' => __( 'Parent Item', 'makeOwnTheme' ),
159     'parent_item_colon' => __( 'Parent Item:', 'makeOwnTheme' ),
160     'new_item_name' => __( 'New Item Name', 'makeOwnTheme' ),
161     'add_new_item' => __( 'Add New Item', 'makeOwnTheme' ),
162     'edit_item' => __( 'Edit Item', 'makeOwnTheme' ),
163     'update_item' => __( 'Update Item', 'makeOwnTheme' ),
164     'view_item' => __( 'View Item', 'makeOwnTheme' ),
165     'separate_items_with_commas' => __( 'Separate items with commas', 'makeOwnTheme' ),
166     'add_or_remove_items' => __( 'Add or remove items', 'makeOwnTheme' ),
167     'choose_from_most_used' => __( 'Choose from the most used', 'makeOwnTheme' ),
168     'popular_items' => __( 'Popular Items', 'makeOwnTheme' ),
169     'search_items' => __( 'Search Items', 'makeOwnTheme' ),
170     'not_found' => __( 'Not Found', 'makeOwnTheme' ),
171     'no_terms' => __( 'No items', 'makeOwnTheme' ),
172     'items_list' => __( 'Items list', 'makeOwnTheme' ),
173     'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),
174 );
175 $args_category = array(
176     'labels' => $labels_category,
177     'hierarchical' => true,
178     'public' => true,
179     'show_ui' => true,
180     'show_admin_column' => true,
181     'show_in_nav_menus' => true,
182     'show_tagcloud' => true,
183     'query var' => true,
```


Les “custom post type” dans le back office



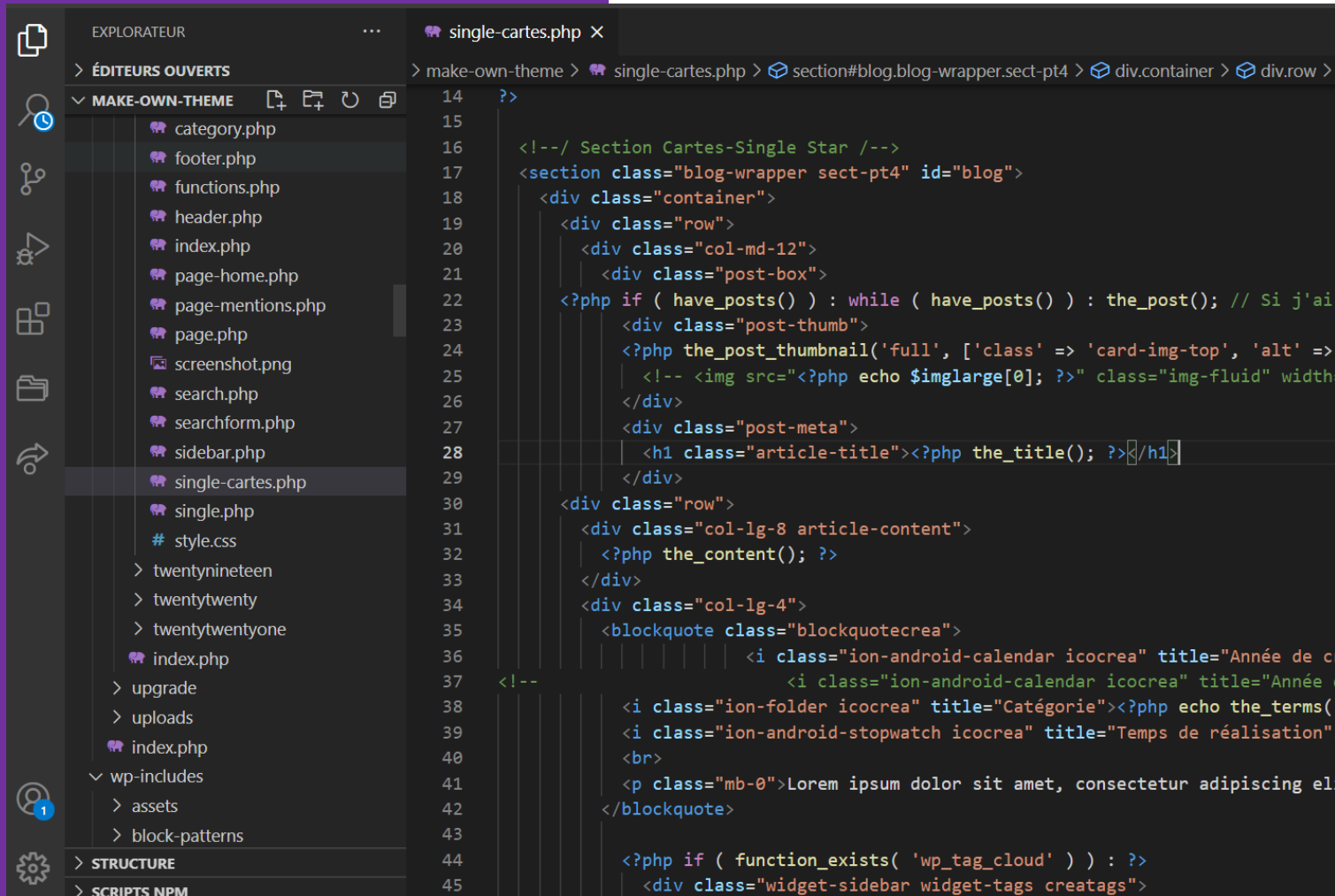
Quand le « custom post type » et ses taxonomies sont déclarés, vous devriez avoir dans votre backoffice un affichage similaire à cette capture.



Wordpress

Création de la page single-SLUG-DU-CPT.php pour l'affichage d'un item de ce post type

Page pour custom post type



```
14  ?>
15
16  <!--/ Section Cartes-Single Star /-->
17  <section class="blog-wrapper sect-pt4" id="blog">
18    <div class="container">
19      <div class="row">
20        <div class="col-md-12">
21          <div class="post-box">
22            <?php if ( have_posts() ) : while ( have_posts() ) : the_post(); // Si j'ai
23              <div class="post-thumb">
24                <?php the_post_thumbnail('full', ['class' => 'card-img-top', 'alt' =>
25                  <!-- 
27                <div class="post-meta">
28                  <h1 class="article-title"><?php the_title(); ?></h1>
29                </div>
30              <div class="row">
31                <div class="col-lg-8 article-content">
32                  <?php the_content(); ?>
33                </div>
34                <div class="col-lg-4">
35                  <blockquote class="blockquotecrea">
36                    <i class="ion-android-calendar icocrea" title="Année de cr
37                  <!--
38                    <i class="ion-android-calendar icocrea" title="Année d
39                    <i class="ion-folder icocrea" title="Catégorie"><?php echo the_terms(
40                    <i class="ion-android-stopwatch icocrea" title="Temps de réalisation">
41                    <br>
42                    <p class="mb-0">Lorem ipsum dolor sit amet, consectetur adipiscing eli
43                  </blockquote>
44                <?php if ( function_exists( 'wp_tag_cloud' ) ) : ?>
45                  <div class="widget-sidebar widget-tags creatags">
```

Comme pour les articles de Wordpress qui ont un affichage spécifique pour chaque articles.

Les « custom post type » en ont un aussi.

Il suffit de créer un fichier à la racine du thème en suivant se nommage « single-cpt.php ».

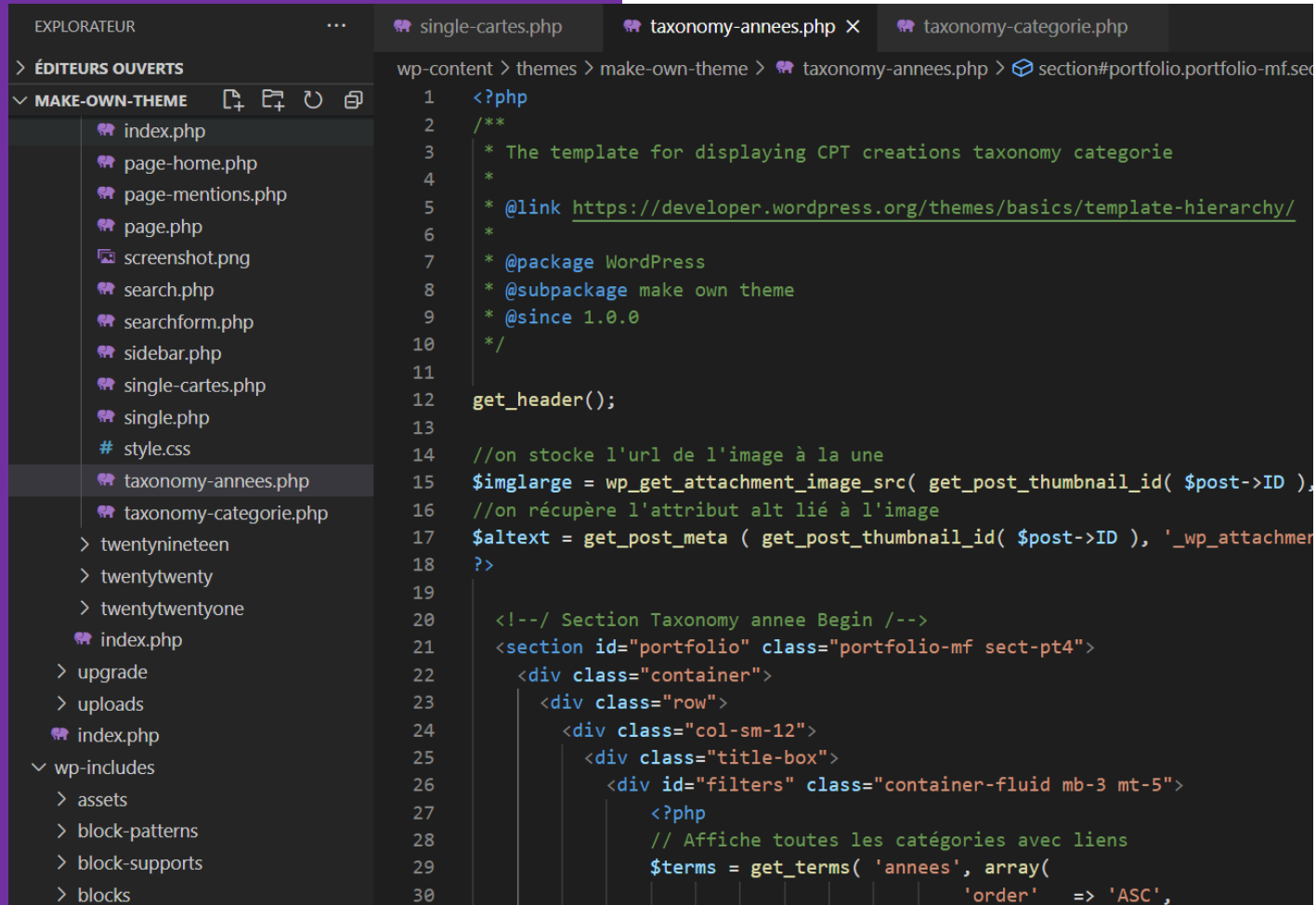
Le terme « cpt » sera a remplacer par le nom du « custom post type ».



Wordpress

Création de la page taxonomy-SLUG-TAXONOMY.php pour l'affichage des post type d'une taxonomie spécifique

Taxonomie “annees”



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer is titled "EXPLORATEUR" and shows a directory structure for a WordPress theme. The code editor is titled "taxonomy-annees.php" and shows the following code:

```
1 <?php
2 /**
3  * The template for displaying CPT creations taxonomy categorie
4  *
5  * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
6  *
7  * @package WordPress
8  * @subpackage make own theme
9  * @since 1.0.0
10 */
11
12 get_header();
13
14 //on stocke l'url de l'image à la une
15 $imlarge = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ),
16 //on récupère l'attribut alt lié à l'image
17 $alttext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attacher
18 ?>
19
20 <!--/ Section Taxonomy annee Begin /-->
21 <section id="portfolio" class="portfolio-mf sect-pt4">
22     <div class="container">
23         <div class="row">
24             <div class="col-sm-12">
25                 <div class="title-box">
26                     <div id="filters" class="container-fluid mb-3 mt-5">
27                         <?php
28                         // Affiche toutes les catégories avec liens
29                         $terms = get_terms( 'annees', array(
30                             'order' => 'ASC',
```

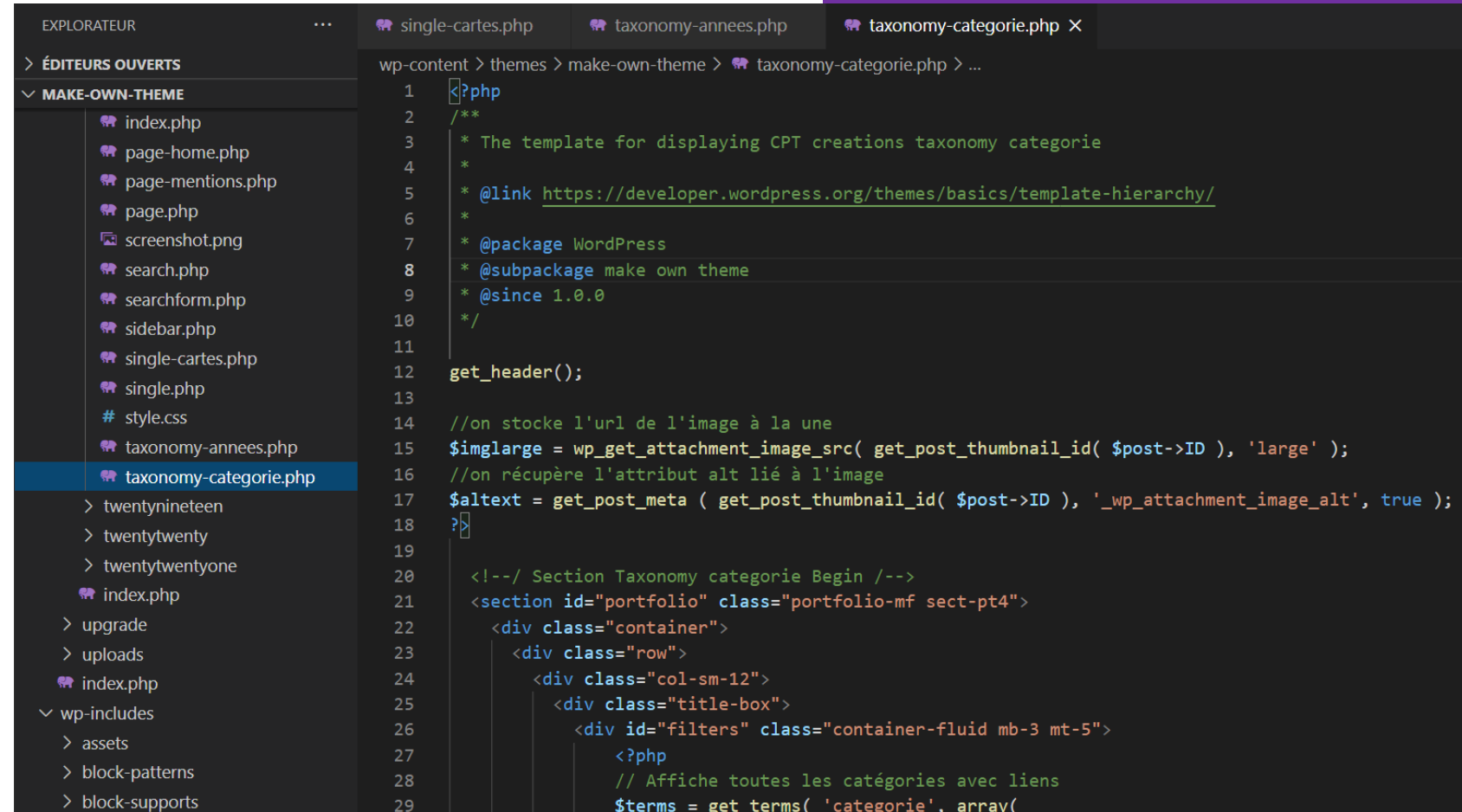
En mettant en place le fichier
« taxonomy-annees.php », il est possible
de gérer une page qui affichera une liste
de « custom post type » qui ont une
taxonomie « années » en particulier.

Taxonomie “categorie”

Tout comme la taxonomie « années », il est possible de créer une page dédiée à la taxonomie « catégorie ».

Il faut créer le fichier « taxonomy-categorie.php » pour y dispatcher l’affichage correspondant.

Dans la requête utilisée pour récupérer les articles correspondant à la taxonomie, il ne faut pas oublier de rajouter un paramètre pour celle-ci.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer is titled 'EXPLORATEUR' and shows a directory structure for 'MAKE-OWN-THEME'. The file 'taxonomy-categorie.php' is selected. The code editor shows the content of 'taxonomy-categorie.php'.

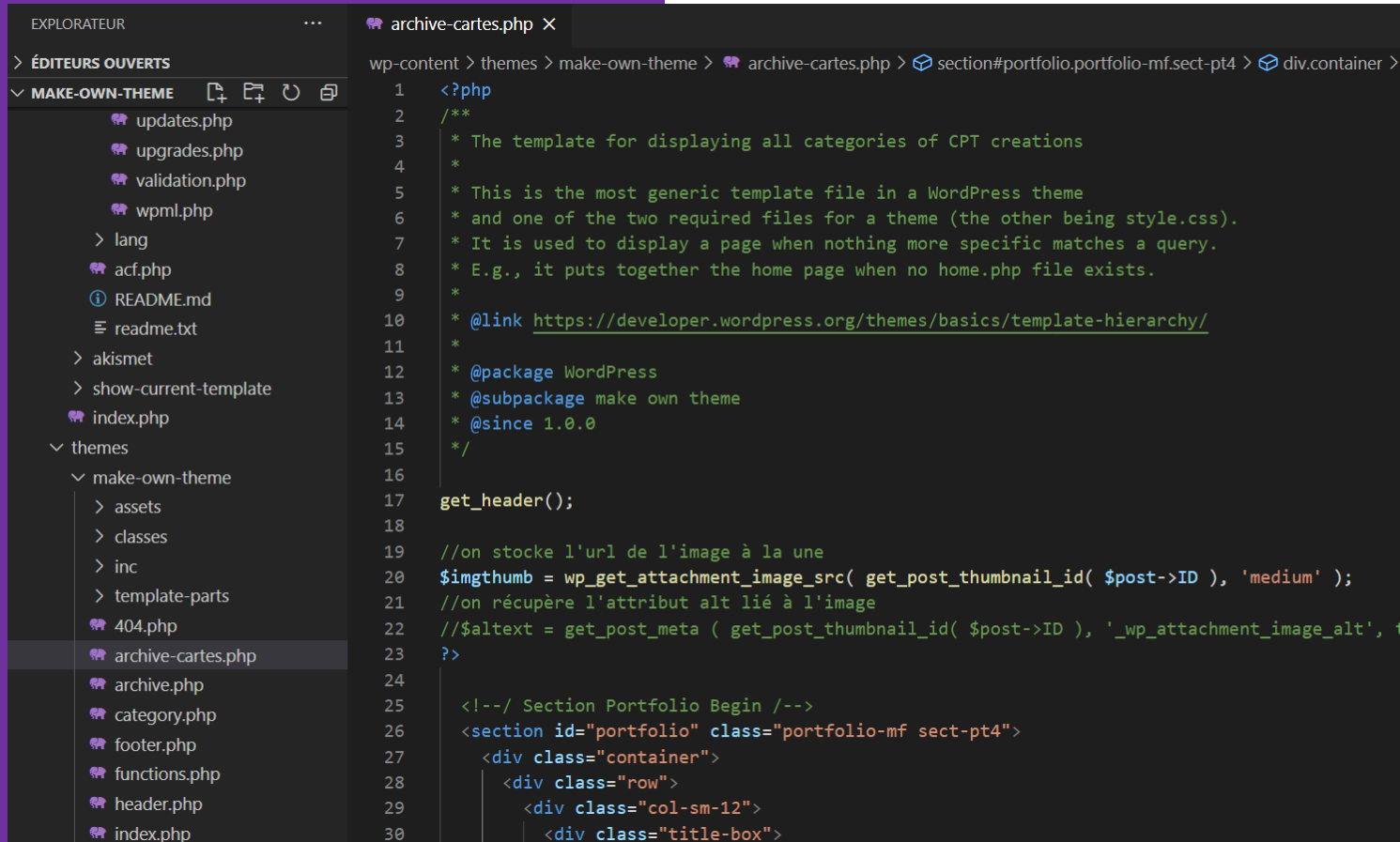
```
wp-content > themes > make-own-theme > taxonomy-categorie.php > ...
1  <?php
2  /**
3   * The template for displaying CPT creations taxonomy categorie
4   *
5   * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
6   *
7   * @package WordPress
8   * @subpackage make own theme
9   * @since 1.0.0
10  */
11
12  get_header();
13
14  //on stocke l'url de l'image à la une
15  $imglarge = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'large' );
16  //on récupère l'attribut alt lié à l'image
17  $alttext = get_post_meta( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
18  ?>
19
20  <!--/ Section Taxonomy categorie Begin /-->
21  <section id="portfolio" class="portfolio-mf sect-pt4">
22    <div class="container">
23      <div class="row">
24        <div class="col-sm-12">
25          <div class="title-box">
26            <div id="filters" class="container-fluid mb-3 mt-5">
27              <?php
28              // Affiche toutes les catégories avec liens
29              $terms = get_terms( 'categorie', array(
```



Wordpress

Création de la page archive-SLUG-CPT.php
pour l'affichage de tous les items du post
type

Page d'archive pour le custom post type



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows the directory structure of a WordPress theme, with 'archive-cartes.php' selected under 'make-own-theme'. The code editor shows the content of 'archive-cartes.php', which is a PHP template file for displaying all categories of CPT creations. The code includes comments, a link to the WordPress theme hierarchy, and a package declaration for WordPress. The code also includes a call to 'get_header()' and a loop to display the content.

```
1 <?php
2 /**
3  * The template for displaying all categories of CPT creations
4  *
5  * This is the most generic template file in a WordPress theme
6  * and one of the two required files for a theme (the other being style.css).
7  * It is used to display a page when nothing more specific matches a query.
8  * E.g., it puts together the home page when no home.php file exists.
9  *
10 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
11 *
12 * @package WordPress
13 * @subpackage make own theme
14 * @since 1.0.0
15 */
16
17 get_header();
18
19 //on stocke l'url de l'image à la une
20 $imgthumb = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'medium' );
21 //on récupère l'attribut alt lié à l'image
22 //$alttext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', t
23 ?>
24
25 <!--/ Section Portfolio Begin /-->
26 <section id="portfolio" class="portfolio-mf sect-pt4">
27     <div class="container">
28         <div class="row">
29             <div class="col-sm-12">
30                 <div class="title-box">
```

Pour mettre en place une page d'archives pour le « custom post type », il faut créer à la racine du thème le fichier « archive-cartes.php ».

Une fois mis en place, il ne faut pas oublier de mettre en place des filtres pour les différentes taxonomies.

Et pour finir, il faut utiliser la boucle de Wordpress pour afficher tous les articles concernés.



Wordpress

Mise en place d'une boucle spécifique permettant le filtrage des items du CPT en fonction des différents « Terms » d'une Taxonomie

Boucle filtrage custom posts en fonction d'une taxonomie

```
$args = array(
    'post_type' => 'cartes',
    'posts_per_page' => 5,
    'orderby' => 'date',
    'order' => 'DESC',
    // 'post_parent' => 0,
    'paged' => $paged,
    'tax_query' => array(
        array(
            'taxonomy' => 'annees',
            'field' => 'slug',
            'terms' => $catslug,
        ),
    ),
);
$loop = new WP_Query( $args );
```

Les « custom posts » peuvent être filtrer par taxonomie directement dans les arguments de la requête mise en place.

Il suffit de le préciser dans un tableau qui servira d'arguments.

Ici dans le « tax_query », c'est la taxonomie « annees » qui est appelée avec une catégorie définie en amont dans la variable « \$catslug ».



Wordpress

Point sur la sécurité dans WordPress



Protéger “wp-admin”

Pour protéger au maximum la page de login vers l'administration de Wordpress, il est plus prudent de changer l'url d'accès vers celle-ci.

Grâce au plugin « change wp-admin login », la configuration se fait rapidement et efficacement.

Voici le lien vers cette ressource:

<https://wordpress.org/plugins/change-wp-admin-login/>

Réglages pour la sécurité

Depuis quelques versions maintenant, Wordpress propose un « état » de la sécurité sur le site. Il est bien sur essentiel de vérifier la sécurité rapidement.

Dans cette optique, il faut toujours vérifier si tous est à jour sur le site, surtout les plugins.

Lors du paramétrage de la base de données dans la configuration, il est important de préfixer celle-ci avec un autre préfixe que « wp ».

Il faut rester vigilant sur la création du mot de passe en préférant utiliser le générateur de mot de passe.





Wordpress

Préparation et migration du thème d'exercice
de son environnement local vers un serveur
de production via phpMyAdmin et FileZilla

Migration base de données

Exportation des tables depuis la base de données « make-own-theme »

Modèles d'exportation :

Nouveau modèle :

Modèles existants :

Modèle :

-- Sélectionner un modèle --

Méthode d'exportation :

- ☐ Rapide, n'afficher qu'un minimum d'options
- ☒ Personnalisée, afficher toutes les options possibles

Format :

SQL

Tables :

	Tables	Structure	Données
	Tout sélectionner	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_commentmeta	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_comments	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_links	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_options	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_postmeta	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_posts	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_termmeta	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	wp_terms	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Sortie :

■ Console de requêtes SQL

Faire Ctrl+Entrée pour exécuter la requête

>

Avant de commencer la migration du site, il faut penser à exporter la base de données.

Allez sur la page suivante « localhost/phpmyadmin » sur votre navigateur puis sur la base de données concernée.

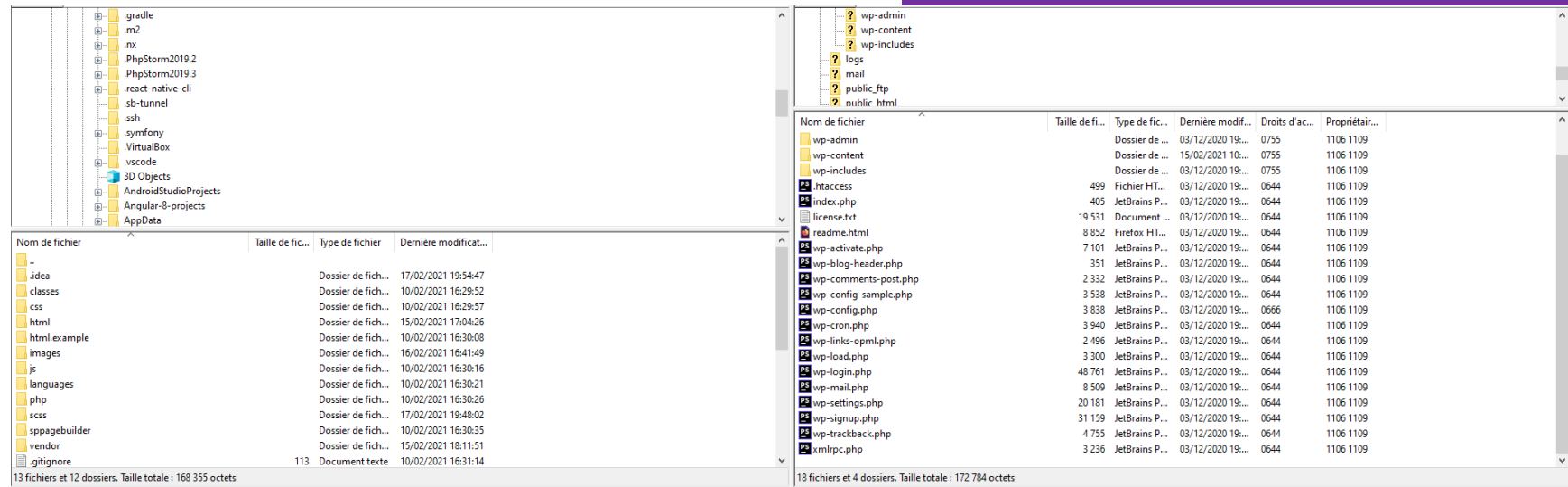
Il vous suffit ensuite de cliquer sur « exporter » pour avoir accès à l'exportation des tables depuis la base de données concernée.

Migration fichiers sources sur le serveur

Les fichiers contenu dans le dossier en local doivent être transférés en ligne dans le dossier qui correspond au site.

Le logiciel « Filezilla » permet de faire ses manipulations sur le serveur par un simple glisser/déposer.

Il ne faut pas oublier ses identifiants pour se connecter sur le serveur via ce logiciel.



Modifications de “wp-config.php” sur le serveur

```
/** Réglages MySQL - Votre hébergeur doit vous  
fournir ces informations. ** //  
/** Nom de la base de données de WordPress. */  
define( 'DB_NAME', 'votre_nom_de_bdd' );  
  
/** Utilisateur de la base de données MySQL. */  
define( 'DB_USER', 'votre_utilisateur_de_bdd' );  
  
/** Mot de passe de la base de données MySQL. */  
define( 'DB_PASSWORD', 'votre_mdp_de_bdd' );  
  
/** Adresse de l'hébergement MySQL. */  
define( 'DB_HOST', 'localhost' );  
  
/** Jeu de caractères à utiliser par la base de  
données lors de la création des tables. */  
define( 'DB_CHARSET', 'utf8' );
```

Il faut aussi changer les paramètres mis en place dans « wp-config.php » car ce sont ceux du site en local.

Pour les changer, il suffit de changer la deuxième valeur mise en paramètres dans les fonctions « define ».

Vous trouverez ses valeurs chez votre hébergeur. Pour la base de données, il faudra la mettre en place sur l'hébergement et récupérer les identifiants lors de la création de celle-ci.

Mettre à jour les urls du site en ligne

Lorsqu'il faut enregistrer le changement des liens suite à un passage entre le site en local vers la production, c'est l'outil « srdb » qui est le plus utile.

Il permet de remplacer en base de données, les anciennes urls par la nouvelle correspondant au site en ligne.

Une fois que les changements sont faits, il faut le supprimer du serveur pour éviter des problèmes de sécurité.

The screenshot shows the srdb search/replace tool interface. It has a red header bar. The main content area is divided into sections: 'search/replace', 'database', 'tables', 'actions', and 'delete'. In the 'search/replace' section, there are input fields for 'replace' (http://www.ancienneurl.) and 'with' (http://www.nouvelleurl.), and a checkbox for 'use regex'. The 'database' section has input fields for 'name' (a blue box), 'user' (root), 'pass' (root), and 'host' (localhost). The 'tables' section has radio buttons for 'all tables' (selected) and 'select tables', and two text boxes for 'columns to exclude (optional, comma separated)' (eg. guid) and 'columns to include only (optional, comma separated)' (eg. post_content, post_excerpt). The 'actions' section has buttons for 'update details', 'dry run', 'live run', 'convert to innodb', and 'convert to utf8 unicode'. The 'delete' section has a 'delete me' button and a note: 'Once you're done click the delete me button to secure your server'.

Les derniers réglages



Maintenant que tout est mis en place sur le serveur, il faut scruter le site en ligne pour vérifier que tout marche correctement.

Il ne faut pas hésiter à tester tous les liens, les affichages, les pages, les ressources, etc...

Et pour terminer, faire un petit tour sur l'administration pour vérifier que tout est en règle comme l'état du site, les contenus déjà créés dans l'ancienne administration.