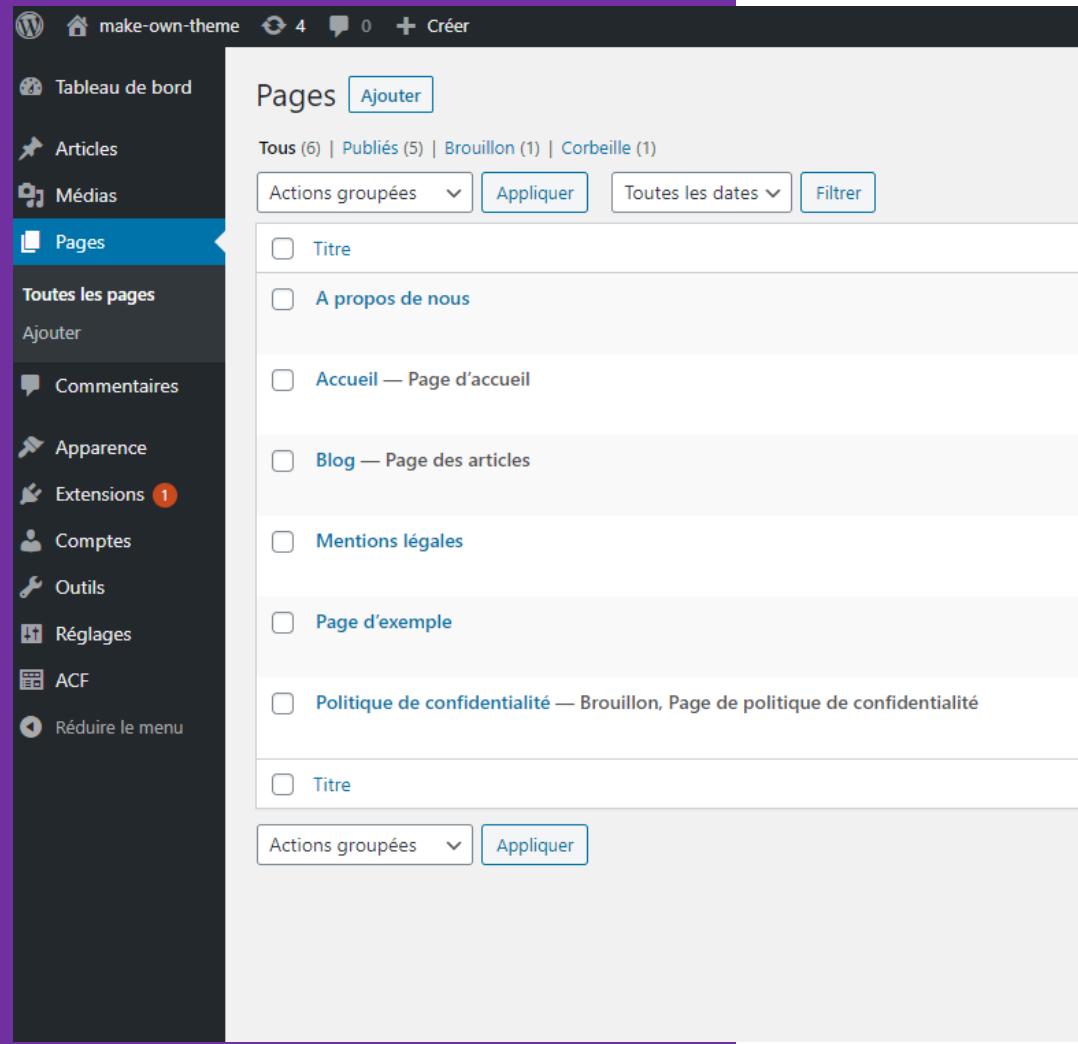




Wordpress

Création de la page des articles (single.php)

Paramétrages avant conception de la page des articles



The screenshot shows the WordPress backoffice interface. The left sidebar has a dark theme with various menu items: Tableau de bord, Articles, Médias, Pages (which is highlighted in blue), Toutes les pages, Ajouter, Commentaires, Apparence, Extensions (with a red notification badge), Comptes, Outils, Réglages, ACF, and Réduire le menu. The main content area is titled 'Pages' with a 'Ajouter' button. It shows a list of pages: Tous (6) | Publié (5) | Brouillon (1) | Corbeille (1). Below this are buttons for 'Actions groupées', 'Appliquer', 'Toutes les dates', and 'Filtrer'. The page list includes: Titre, A propos de nous, Accueil — Page d'accueil, Blog — Page des articles (which is selected and highlighted in blue), Mentions légales, Page d'exemple, and Politique de confidentialité — Brouillon, Page de politique de confidentialité. At the bottom are buttons for 'Actions groupées' and 'Appliquer'.

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.

The screenshot shows the WordPress admin dashboard with the 'make-own-theme' theme selected. The left sidebar has a dark background with white text, showing various menu items like 'Tableau de bord', 'Articles', 'Médias', 'Pages', etc., with 'Réglages' being the active section. The main content area is titled 'Réglages de lecture'. It contains several configuration options:

- La page d'accueil affiche**: A radio button group where 'Une page statique (choisir ci-dessous)' is selected, while 'Les derniers articles' is unselected.
- Page d'accueil :** A dropdown menu set to 'Accueil'.
- Page des articles :** A dropdown menu set to 'Blog'.
- Les pages du site doivent afficher au plus**: A text input field containing '5' followed by 'articles'.
- Les flux de syndication affichent les derniers**: A text input field containing '10' followed by 'éléments'.
- Dans chaque publication du flux, inclure**: A radio button group where 'Le texte complet' is selected, while 'L'extrait' is unselected.
- Votre thème détermine comment le contenu est affiché dans les navi**: A descriptive note about theme-specific content display.
- Visibilité par les moteurs de recherche**: A checkbox labeled 'Demander aux moteurs de recherche de ne pas indexer ce site', which is checked.
- Certains moteurs de recherche peuvent décider de l'indexer malgré to**: A descriptive note about search engine indexing decisions.

At the bottom right of the content area is a blue button labeled 'Enregistrer les modifications'.

Page pour le blog avec “index.php”

The screenshot shows a code editor interface with several tabs at the top: index.php, page-home.php, header.php, main-nav.php, and style.css. The index.php tab is active. The left sidebar shows a tree view of files under 'MAKE-OWN-THEME' and 'STRUCTURE'. The main pane displays the PHP code for the index.php file, which includes HTML structure for a blog homepage, a loop for displaying posts, and conditional logic for handling empty categories and no posts.

```
<?php get_header() ?>
<h1 class="text-center mt-5 mb-5"><?= get_the_title(get_option('page_for_posts')) ?></h1>
<div class="container mt-5 mb-5">
    <div class="row">
        <div class="col-sm-12 col-md-8">
            <?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-subt
<?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
                                </div>
                            </div>
                        <?php endwhile ?>
                    <div class="col-12 text-center">
                        <div class="pagination">
                            <?php paginate_links() ?>
                        </div>
                    </div>
                <?php else: ?>
                    <h1>Il n'y a pas encore d'actualités.</h1>
                <?php endif ?>
```

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

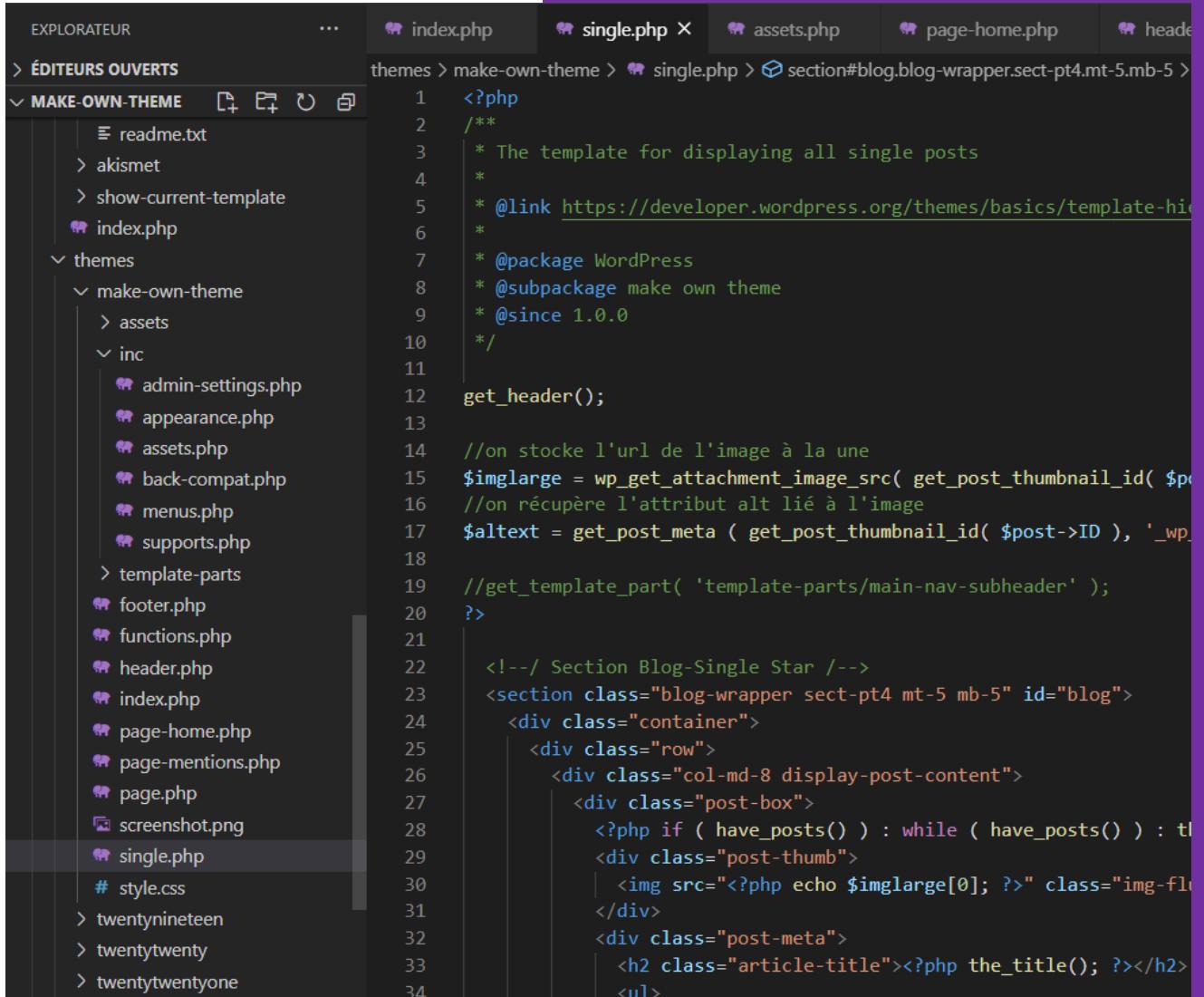
C'est donc dans ce fichier qu'il faudra gérer l'affichage des articles grâce à un système de boucle.

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 several files open in tabs at the top: index.php, single.php (which is the active tab), assets.php, page-home.php, and header.php. The left pane is an 'EXPLORATEUR' (File Explorer) showing the directory structure of the theme 'make-own-theme'. The 'single.php' file is located in the 'inc' folder. The code in 'single.php' is as follows:

```
<?php
/*
 * The template for displaying all single posts
 *
 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
 *
 * @package WordPress
 * @subpackage make_own_theme
 * @since 1.0.0
 */
get_header();

//on stocke l'url de l'image à la une
$imglarge = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'large' );
//on récupère l'attribut alt lié à l'image
$alttext = get_post_meta( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
//get_template_part( 'template-parts/main-nav-subheader' );
?>


<section class="blog-wrapper sect-pt4 mt-5 mb-5" id="blog">
    <div class="container">
        <div class="row">
            <div class="col-md-8 display-post-content">
                <div class="post-box">
                    <?php if ( have_posts() ) : while ( have_posts() ) : the_post() ; ?>
                    <div class="post-thumb">
                        | 
                    </div>
                    <div class="post-meta">
                        <h2 class="article-title"><?php the_title(); ?></h2>
                        <ul class="list-group list-style-type-none">
                            <li><?php the_category(); ?></li>
                            <li><?php the_date(); ?></li>
                            <li><?php the_author(); ?></li>
                        </ul>
                    </div>
                    <div class="post-content">
                        <p><?php the_content(); ?></p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
```



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 interface with the following details:

- EXPLORATEUR**: Shows the project structure under "MAKE-OWN-THEME".
- ÉDITEURS OUVERTS**: Shows three tabs: "functions.php" (active), "widgets.php", and "assets.php".
- functions.php Content**:

```
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.

The screenshot shows a code editor interface with several tabs at the top: functions.php (selected), widgets.php (closed), assets.php (closed), and page-home.php (closed). The main area displays the contents of the functions.php file:

```
<?php  
add_action('after_setup_theme', function(){  
    register_sidebar([  
        'id' => 'blog-search',  
        'name' => __('Search sidebar', 'makeOwnTheme'),  
        'before_title' => '<div class="sidebar_title">',  
        'after_title' => '</div>',  
        'before_widget' => '<div class="sidebar_widget">',  
        'after_widget' => '</div>'  
    ]);  
});|
```

The left side of the interface shows a file explorer with the following structure:

- EXPLORATEUR
- ÉDITEURS OUVERTS
- MAKE-OWN-THEME
 - 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
 - inc
 - admin-settings.php
 - appearance.php
 - assets.php
 - back-compat.php
 - menus.php
 - supports.php
 - widgets.php

Zone de widget dans le backoffice

The screenshot shows the WordPress backoffice interface. The left sidebar is dark blue with white text, showing navigation links like 'Tableau de bord', 'Articles', 'Médias', 'Pages', 'Commentaires', and 'Apparence'. The 'Apparence' link is highlighted in blue. The main content area has a light gray background. At the top, it says 'Widgets' and 'Gérer avec la prévisualisation en direct'. Below this, a section titled 'Widgets disponibles' lists various widget types with their descriptions and dropdown menus for selection. To the right, there is a sidebar with a title 'Search sidebar'.

Widget Type	Description
Archives	Une archive mensuelle des articles de votre site.
Articles récents	Les articles les plus récents de votre site.
Calendrier	Un calendrier des articles de votre site.
Catégories	Une liste ou un menu déroulant des catégories.
Commentaires récents	Les commentaires les plus récents de votre site.
Flux	Les entrées depuis n'importe quel flux RSS ou Atom.
Galerie	Affiche une galerie d'images.
HTML personnalisé	Code HTML arbitraire.
Image	Affiche une image.
Menu de navigation	Ajoute un menu de navigation dans votre colonne latérale.
Méta	Connexion, flux RSS et liens WordPress.org.
Nuage d'étiquettes	Un nuage de vos étiquettes les plus utilisées.

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 admin interface. On the left, a sidebar menu is open, with 'Widgets' selected. The main area is titled 'Widgets' with a sub-link 'Gérer avec la prévisualisation en direct'. Below this, the heading 'Widgets disponibles' is followed by a list of 20 different widget types, each with a brief description. To the right, there are three panels for configuring specific widgets:

- Search sidebar:** A panel for searching sidebar configurations. It has a 'Rechercher' input field and 'Supprimer' and 'Enregistrer' buttons.
- Catégories:** A panel for configuring the 'Catégories' widget. It includes a 'Titre:' input field ('Catégories'), three checkboxes for display options ('Afficher comme liste déroulante', 'Afficher le nombre d'articles', 'Afficher la hiérarchie'), and 'Supprimer' and 'Enregistrer' buttons.
- Articles récents:** A panel for configuring the 'Articles récents' widget. It includes a 'Titre:' input field ('Les derniers articles'), a 'Nombre d'articles à afficher:' input field (set to 5), a checkbox for 'Afficher la date de la publication?', and 'Supprimer' and 'Enregistrer' buttons.



Wordpress

Création et inclusion du fichier sidebar.php

Sans “sidebar.php”

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

```
,div>


<?php dynamic_sidebar('blog-search') ?>
    <!-- <?php //get_sidebar('blog-search') ?> -->
</div>


```

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.

Avec “sidebar.php”

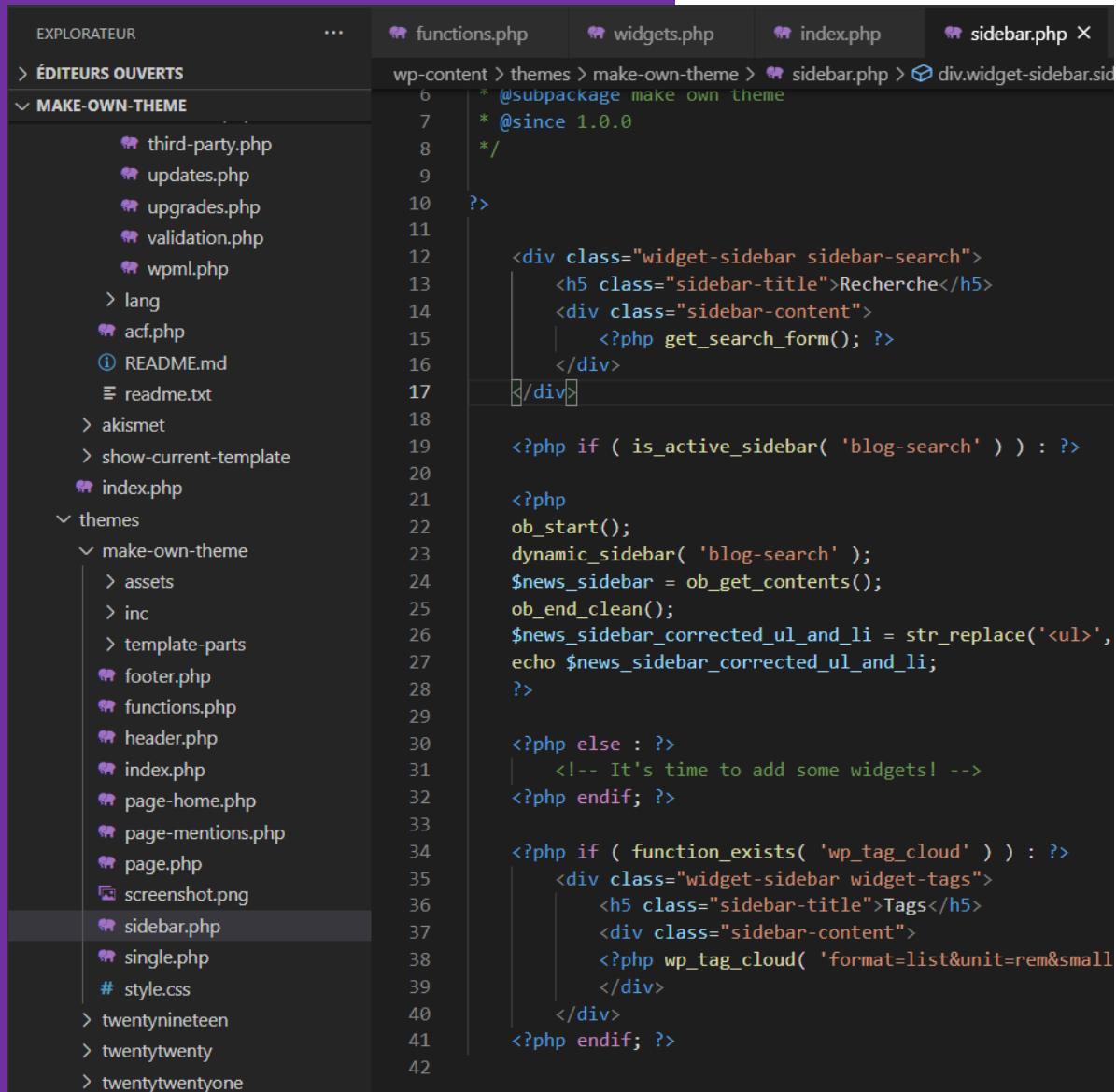
Tout comme la fonction « dynamic_sidebar », la fonction « get_sidebar » permet de récupérer une sidebar lorsqu'elle est spécifiée dans les paramètres.

Si il n'y est mis aucun paramètre, 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>
<br>
```

Avec “sidebar.php”



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar titled "EXPLORATEUR" containing a tree view of files and folders. The "ÉDITEURS OUVERTS" section shows the "functions.php", "widgets.php", "index.php", and "sidebar.php" files. The "MAKE-OWN-THEME" folder is expanded, showing 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", and several "themes" sub-folders. The "make-own-theme" folder is also expanded, showing "assets", "inc", "template-parts", "footer.php", "functions.php", "header.php", "index.php", "page-home.php", "page-mentions.php", "page.php", and a "screenshot.png" file. The "sidebar.php" file is currently selected and highlighted in the sidebar.

```
wp-content > themes > make-own-theme > sidebar.php > div.widget-sidebar.sid
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; ?>
```

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 sur 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>
```

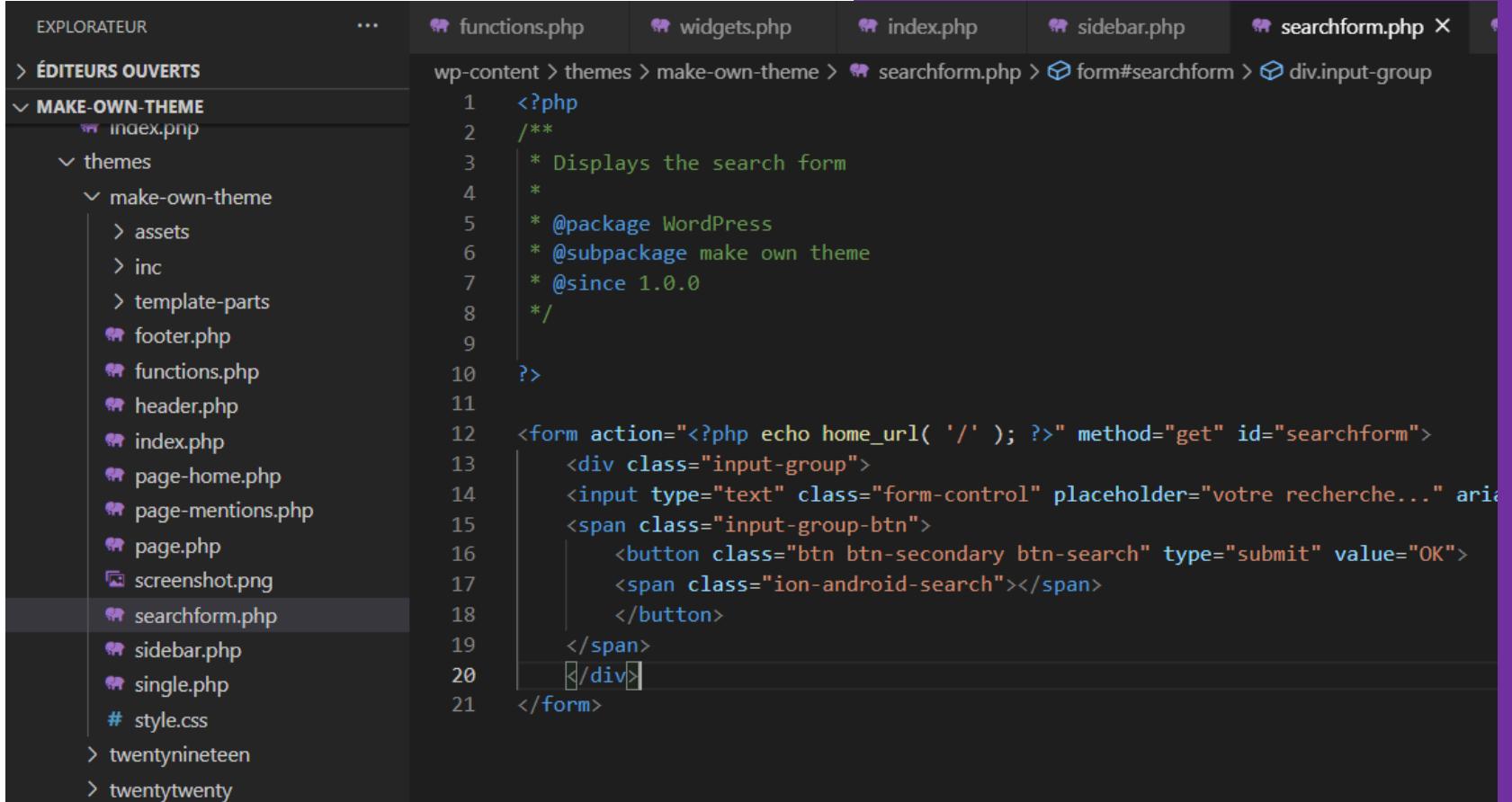
Dans la sidebar ou ailleurs sur le site, il est possible de mettre en place un formulaire de recherche.

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 interface with a dark theme. On the left, there's an 'EXPLORATEUR' sidebar titled 'ÉDITEURS OUVERTS' which lists files in the 'MAKE-OWN-THEME' folder, including 'index.php', 'functions.php', 'header.php', 'index.php', 'page-home.php', 'page-mentions.php', 'page.php', 'screenshot.png', 'searchform.php' (which is currently selected), 'sidebar.php', 'single.php', '# style.css', 'twentynineteen', and 'twentytwenty'. The main panel displays the contents of 'searchform.php'. The code is as follows:

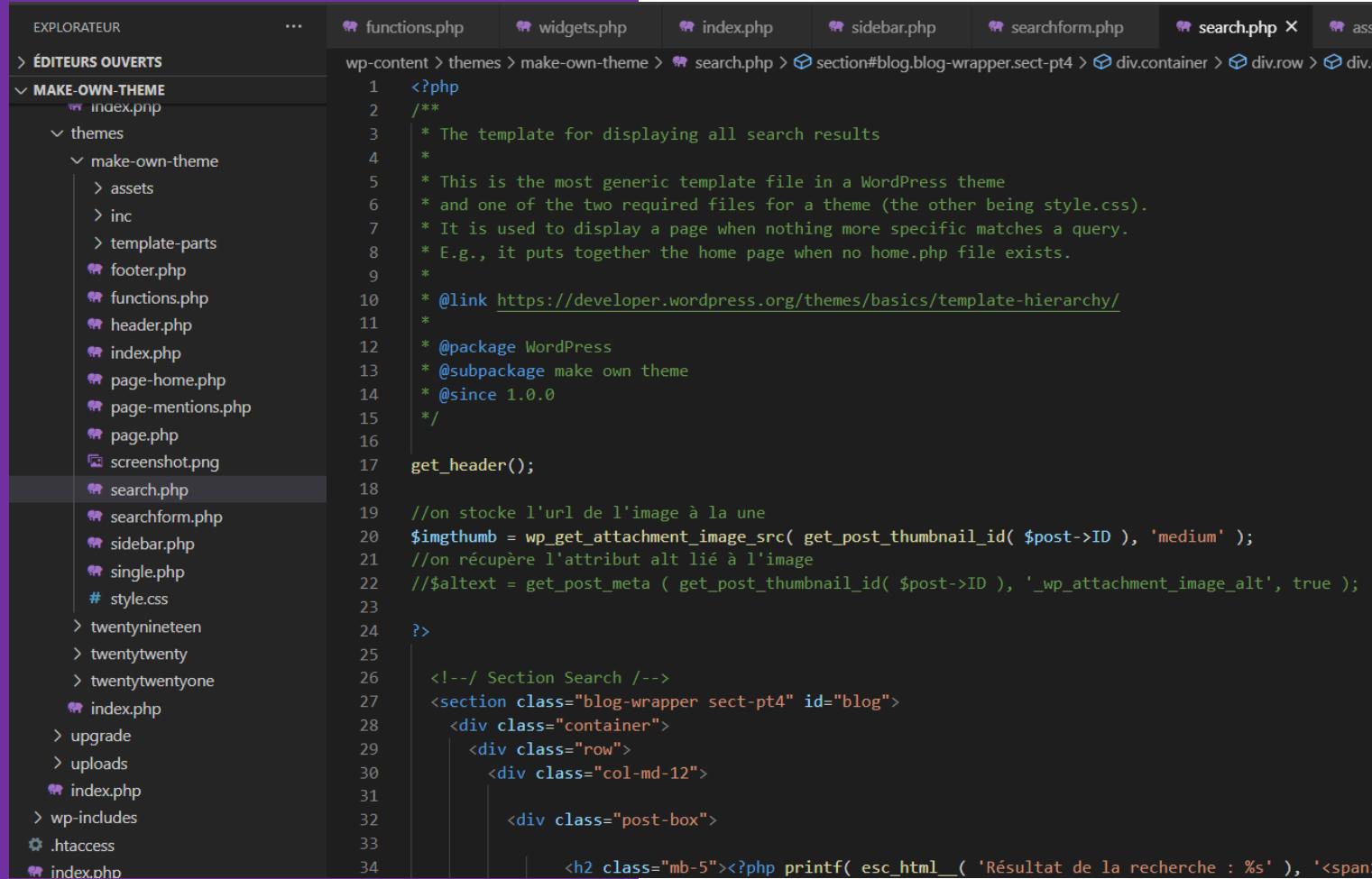
```
wp-content > themes > make-own-theme > searchform.php > form#searchform > div.input-group
1  <?php
2  /**
3   * Displays the search form
4   *
5   * @package WordPress
6   * @subpackage make own theme
7   * @since 1.0.0
8   */
9
10 <?
11
12 <form action=<?php echo home_url( '/' ); ?> method="get" id="searchform">
13     <div class="input-group">
14         <input type="text" class="form-control" placeholder="votre recherche..." aria-
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 interface with a dark theme. On the left is a sidebar titled "EXPLORATEUR" containing a tree view of files and folders. The "MAKE-OWN-THEME" folder is expanded, showing its contents. The "search.php" file is selected and highlighted in the sidebar. The main pane displays the PHP code for the "search.php" file.

```
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( $post->ID ), 'medium' );
21 //on récupère l'attribut alt lié à l'image
22 //$altext = 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                 <div class="post-box">
32                     <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>
            <?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 à 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 ».

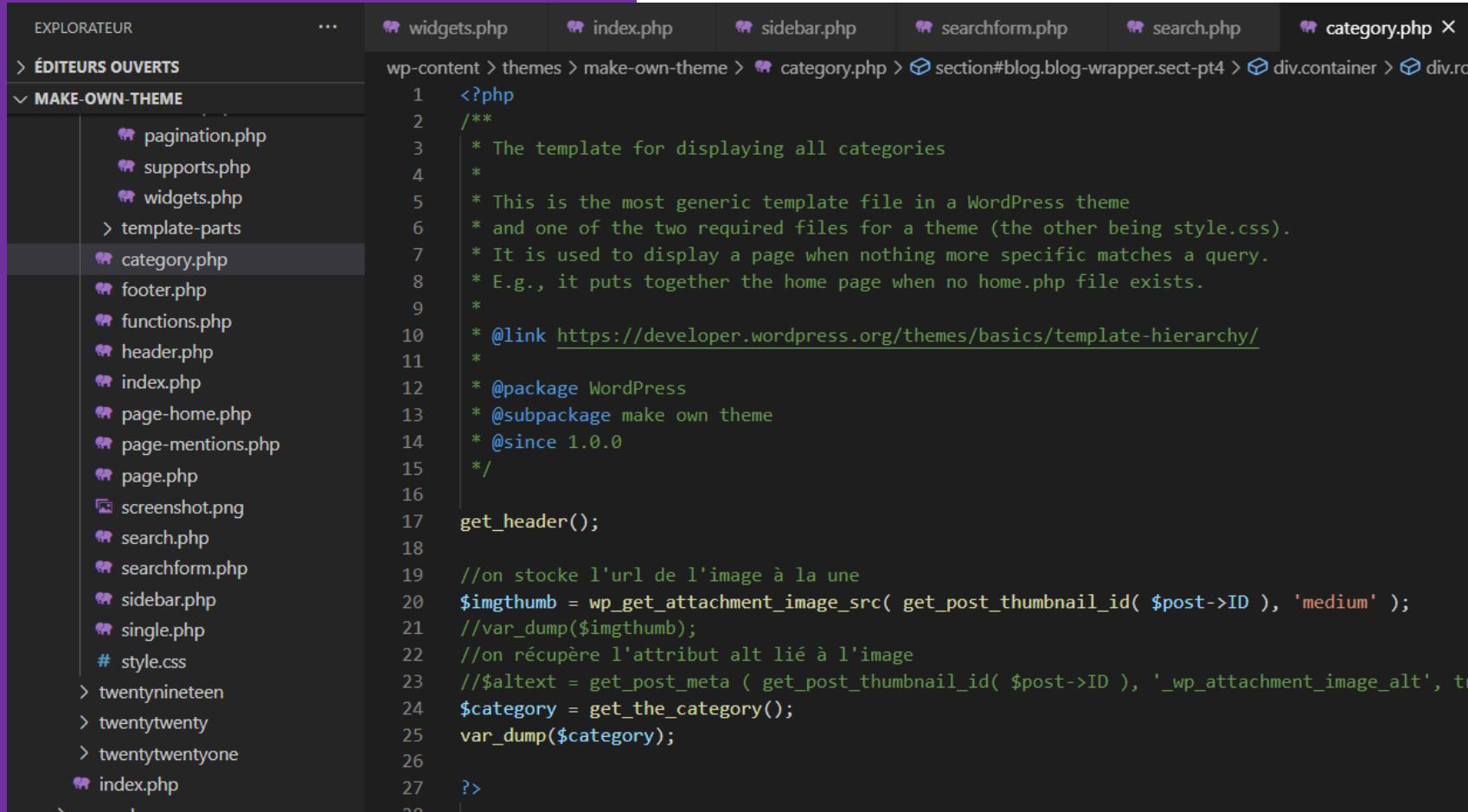
```
<?php while (have_posts()): the_post(); ?>
    <div class="col-sm-12 col-md-12 col-lg-6">
        <div class="card mt-3 mb-3" style="width: 100%">
            <?php the_post_thumbnail('medium', ['class' => 'card-img-top', 'alt' => '']); ?>
            <div class="card-body">
                <h3 class="card-title"><?= 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 the file "category.php" open. The file is located in the "make-own-theme" folder under "wp-content > themes". The code is a template for displaying all categories in a WordPress theme. It includes comments explaining its purpose, links to developer documentation, and variable declarations. The code also includes functions like get_header() and wp_get_attachment_image_src(). A yellow bar at the top of the code editor highlights the current file.

```
EXPLORATEUR ... widgets.php index.php sidebar.php searchform.php search.php category.php X  
wp-content > themes > make-own-theme > category.php > section#blog.blog-wrapper.sect-pt4 > div.container > div.row  
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  
get_header();  
17  
//on stocke l'url de l'image à la une  
18 $imgthumb = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'medium' );  
19 //var_dump($imgthumb);  
20 //on récupère l'attribut alt lié à l'image  
21 //$altext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );  
22 $category = get_the_category();  
23 var_dump($category);  
24  
25 ?>
```

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.

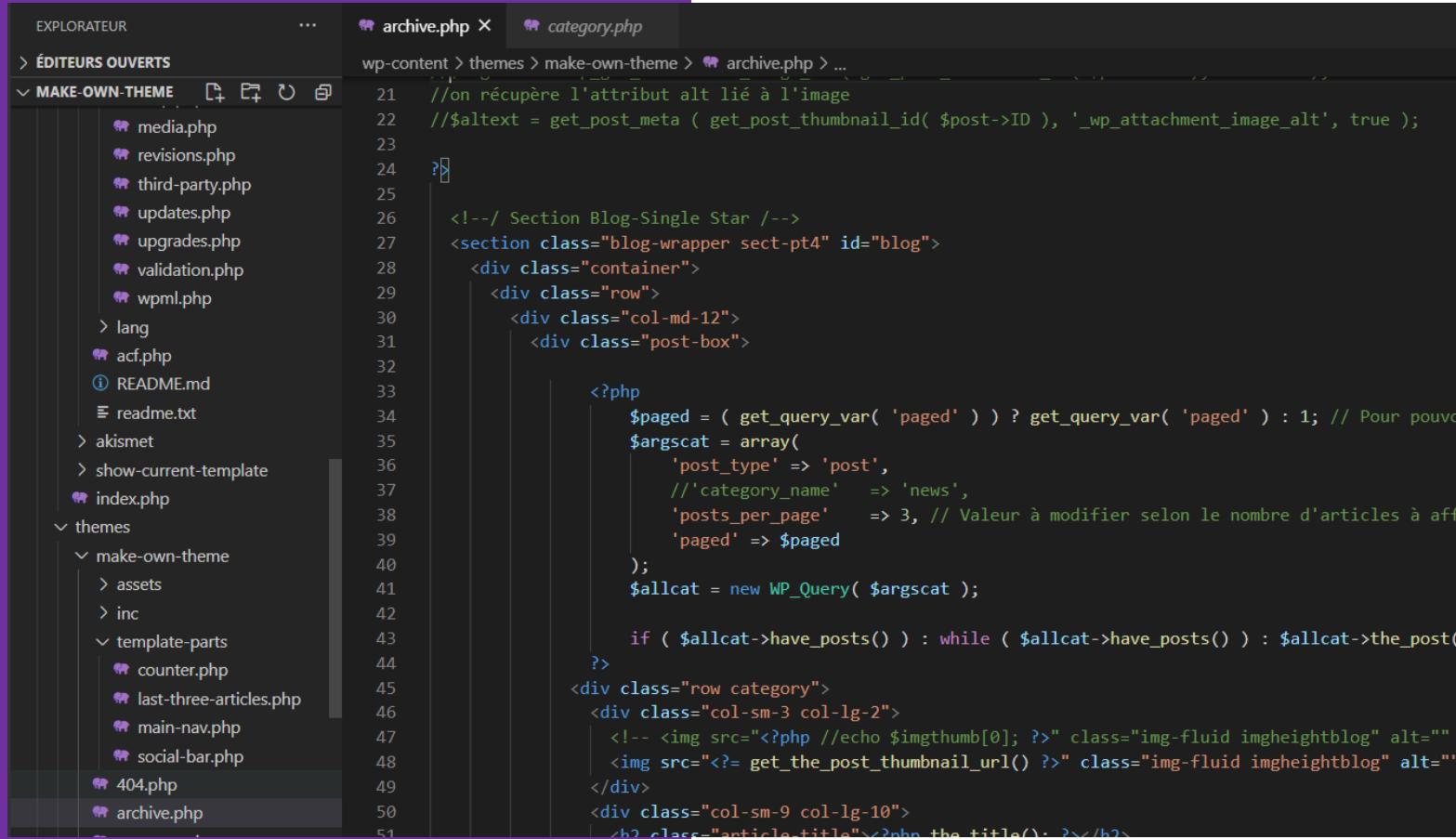
Ensuite, il ne reste plus qu'à mettre en place l'affichage si des articles existent bien dans la catégorie.



Wordpress

Création de la page archive.php

Archive.php



The screenshot shows a code editor interface with a dark theme. On the left, there's an 'EXPLORATEUR' sidebar listing files and folders. In the main area, two tabs are open: 'archive.php' (which is the active tab) and 'category.php'. The code editor displays the PHP code for the 'archive.php' template. The code includes HTML structure for a blog archive, such as sections for 'Blog-Single Star' and 'Category', and a loop to display posts. It also includes logic to handle pagination and query arguments.

```
<!-- Section Blog-Single Star -->
<section class="blog-wrapper sect-pt4" id="blog">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <div class="post-box">

                    <?php
                        $paged = ( get_query_var( 'paged' ) ) ? get_query_var( 'paged' ) : 1; // Pour pouvoir faire une recherche par page
                        $argscat = array(
                            'post_type' => 'post',
                            //category_name' => 'news',
                            'posts_per_page' => 3, // Valeur à modifier selon le nombre d'articles à afficher
                            'paged' => $paged
                        );
                        $allcat = new WP_Query( $argscat );

                        if ( $allcat->have_posts() ) : while ( $allcat->have_posts() ) : $allcat->the_post()
                    ?>
                <div class="row category">
                    <div class="col-sm-3 col-lg-2">
                        <!-- <img src=<?php //echo $imgthumb[0]; ?>" class="img-fluid imgheightblog" alt="" >
                        
                    </div>
                    <div class="col-sm-9 col-lg-10">
                        <b><?php the_title(); ?></b>
```

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

EXPLORATEUR ...

> ÉDITEURS OUVERTS

MAKE-OWN-THEME

- third-party.php
- updates.php
- upgrades.php
- validation.php
- wpml.php

> lang

- acf.php

① 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

404.php

404.php X main-nav.php

wp-content > themes > make-own-theme > 404.php > div.cont

```
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">Pa
11                <hr class="hr">
12            </div>
13            <p class="text-center">
14                Cette page n'existe pas.
15                Vous pouvez retourner à l'
16            </p>
17            <div class="text-center">
18                <a class="btn btn-blue" href=
19            </div>
20        </div>
21    </div>
22<?php get_footer() ?>
```

404.php

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

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

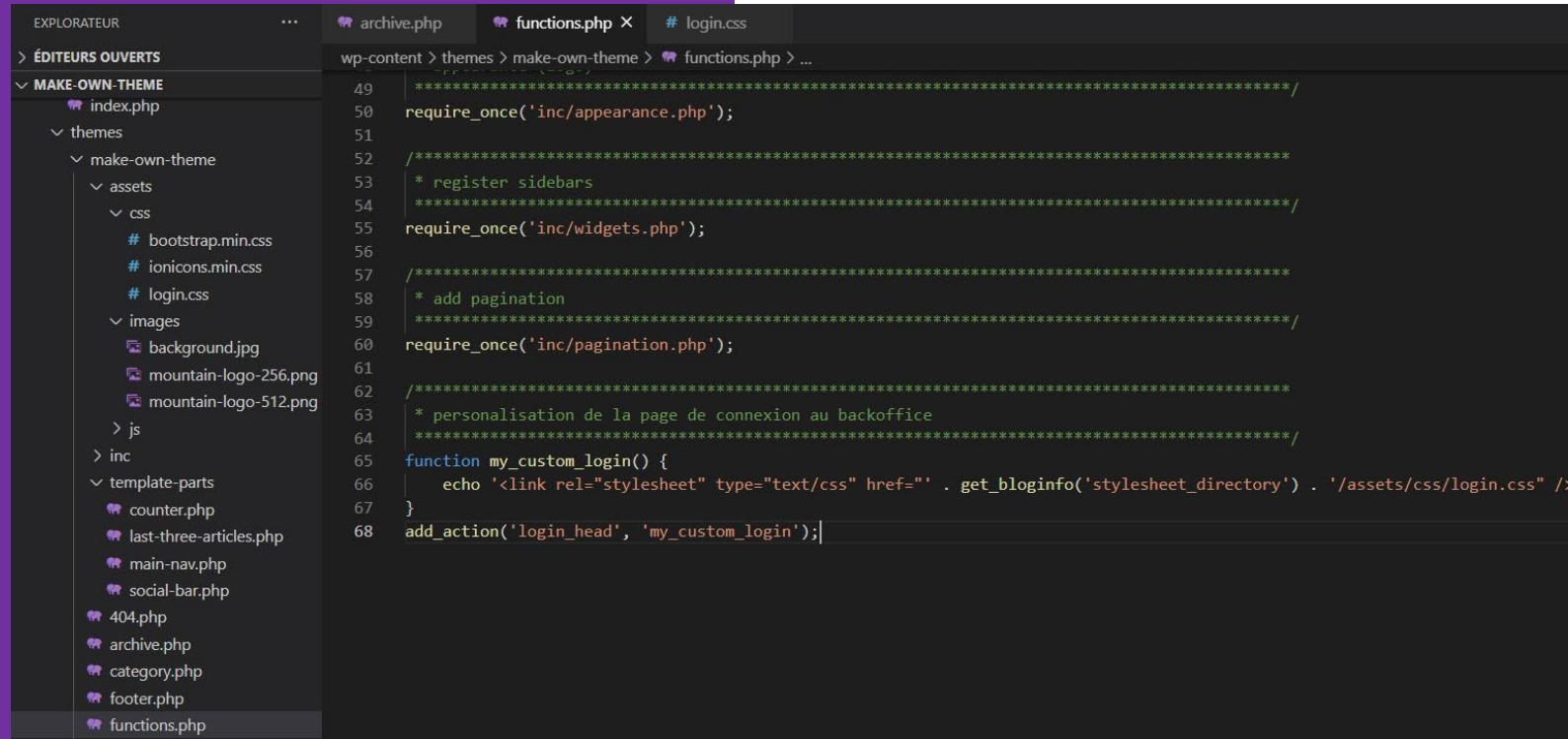
Dans ce fichier, il faut en général mettre un texte explicatif pour prévenir l'utilisateur et un bouton pour qu'il puisse revenir à la page d'accueil.



Wordpress

Personnalisation de la page login.php

Login.php



The screenshot shows a code editor interface with the following details:

- EXPLORATEUR**: Shows the file structure of the theme:
 - MAKE-OWN-THEME
 - index.php
 - themes/make-own-theme/assets/css/bootstrap.min.css
 - themes/make-own-theme/assets/css/ionicons.min.css
 - themes/make-own-theme/assets/css/login.css
 - images/background.jpg
 - images/mountain-logo-256.png
 - images/mountain-logo-512.png
 - js
 - inc
 - template-parts/counter.php
 - template-parts/last-three-articles.php
 - template-parts/main-nav.php
 - template-parts/social-bar.php
 - 404.php
 - archive.php
 - category.php
 - footer.php
 - functions.php
- ÉDITEURS OUVERTS**: Shows the contents of the functions.php file.
- Tabs**: archive.php, functions.php (active), login.css

```
wp-content > themes > make-own-theme > functions.php > ...  
49 | ****  
50 | require_once('inc/appearance.php');  
51 | ****  
52 | * register sidebars  
53 | ****  
54 | require_once('inc/widgets.php');  
55 | ****  
56 | * add pagination  
57 | ****  
58 | * personalisation de la page de connexion au backoffice  
59 | ****  
60 | require_once('inc/pagination.php');  
61 | ****  
62 | * personalisation de la page de connexion au backoffice  
63 | ****  
64 | function my_custom_login()  
65 |     echo '<link rel="stylesheet" type="text/css" href="' . get_bloginfo('stylesheet_directory') . '/assets/css/login.css" />';  
66 | }  
67 | 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 interface with two tabs: 'functions.php' and '# login.css'. The left pane is an 'EXPLORATEUR' (File Explorer) showing the directory structure of a theme named 'make-own-theme'. The 'assets/css' folder contains three files: 'bootstrap.min.css', 'ionicons.min.css', and 'login.css'. The 'login.css' file is currently selected and shown in the right pane. The code in 'login.css' is as follows:

```
/*
 * Custom Login Page CSS
 */

body.login {
    background-image: url('../images/background.jpg');
    /* background-image: linear-gradient(60deg, #C835F4, #F0A5D0); */
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: center;
    background-size: cover;
}

/* Logo */
.login h1 a {
    background: url('../images/mountain-logo-256.png') no-repeat;
    background-size: contain !important;
}

/* formulaire de connexion */
.login form {
    background: #rgba(255, 255, 255, 0.86);
    border-radius: 2px;
}

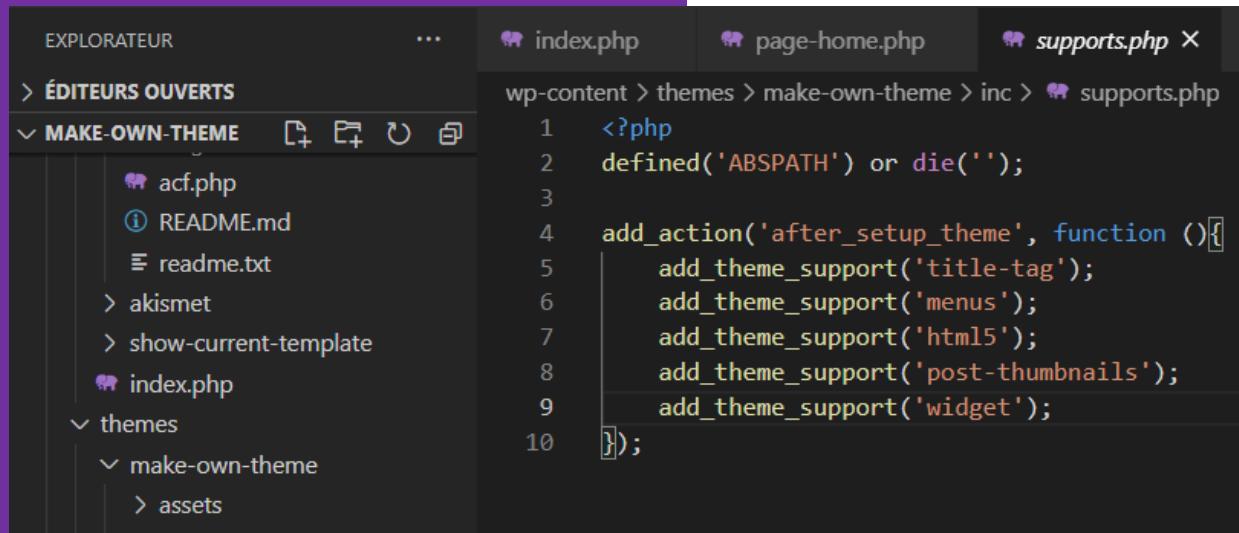
.login label {
    font-size: 12px;
    color: #555555;
}
```



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 the following interface elements:

- EXPLORATEUR**: Shows the directory structure of the theme.
- ÉDITEURS OUVERTS**: Shows three files: index.php, page-home.php, and supports.php.
- MAKE-OWN-THEME**: A toolbar with icons for file operations.

The **supports.php** file content is as follows:

```
<?php
defined('ABSPATH') or die('');

add_action('after_setup_theme', function (){
    add_theme_support('title-tag');
    add_theme_support('menus');
    add_theme_support('html5');
    add_theme_support('post-thumbnails');
    add_theme_support('widget');
});

});
```

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 sur que les ajouts seront faits.

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

Création du menu dans le backoffice

The screenshot shows the WordPress backoffice interface. The left sidebar is dark blue with white icons and text, showing options like 'Tableau de bord', 'Articles', 'Médias', 'Pages', 'Commentaires', and 'Apparence'. The 'Apparence' section is currently selected. The main content area has a light gray background. At the top, it says 'Gérer avec la prévisualisation en direct' and 'navigation principale a été mis à jour.' Below this, there's a button 'Modifier les menus'. A message says 'Modifiez votre menu ci-dessous, ou [créez un nouveau menu](#). N'oubliez pas d'enregistrer vos modifications !'. On the left, there's a sidebar titled 'Ajouter des éléments de menu' with sections for 'Pages', 'Articles', 'Liens personnalisés', and 'Catégories'. Under 'Pages', there are checkboxes for 'Mentions légales', 'A propos de nous', 'Page d'exemple', and 'Tout sélectionner', with a blue 'Ajouter au menu' button. In the center, there's a 'Structure du menu' section with a title 'Nom du menu: navigation principale' and a blue 'Enregistrer le menu' button. Below it, there's a list of menu items: 'Accueil' (under 'Page d'accueil') and 'A propos de nous' (under 'Page'). At the bottom, there's a 'Réglages du menu' section with checkboxes for 'Ajoutez automatiquement des pages' and 'Afficher l'emplacement', and a red 'Supprimer le menu' button.

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

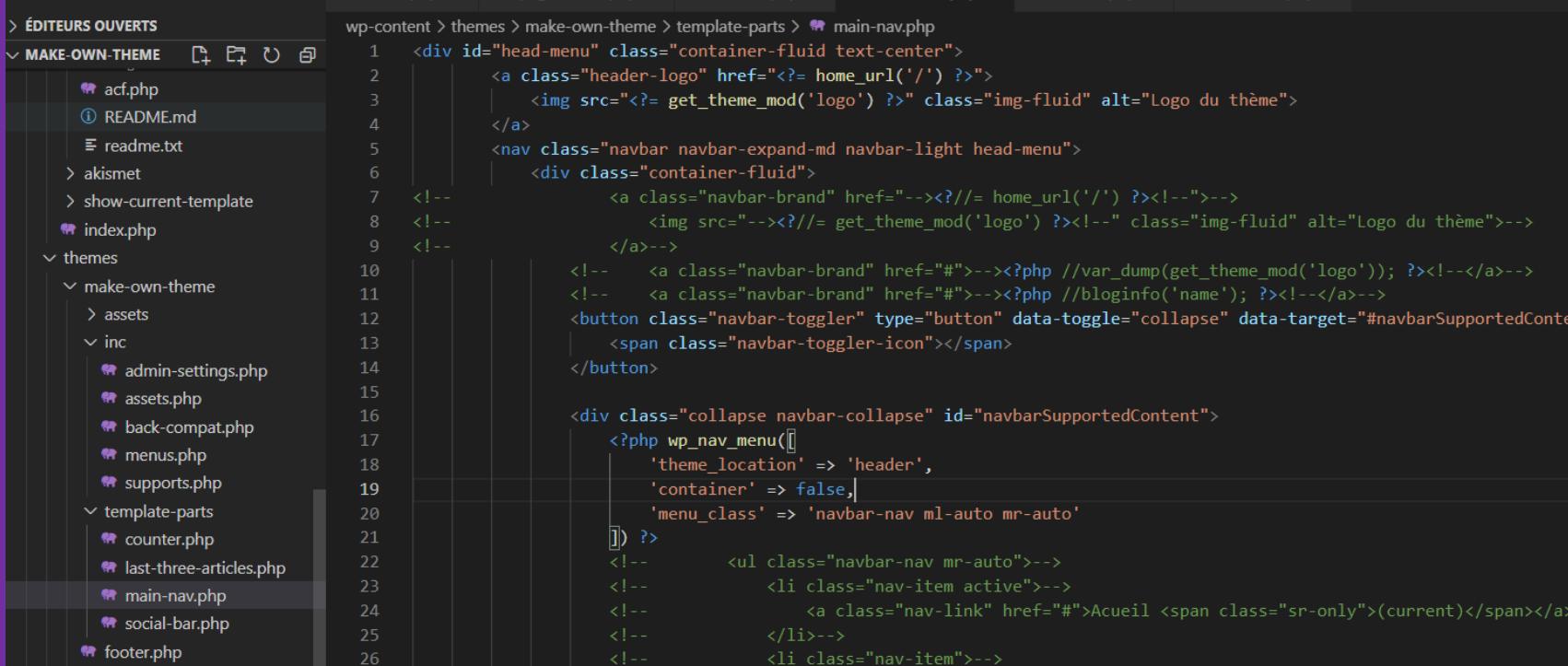
Intégration d'un menu simple

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 the following details:

- ÉDITEURS OUVERTS**: A sidebar on the left.
- MAKE-OWN-THEME**: A folder containing several files: acf.php, README.md, readme.txt, akismet, show-current-template, index.php, and a themes folder.
- themes**: A folder containing make-own-theme, assets, inc, template-parts, and footer.php.
- make-own-theme**: A folder containing assets, inc, and template-parts.
- inc**: A folder containing admin-settings.php, assets.php, back-compat.php, menus.php, supports.php.
- template-parts**: A folder containing counter.php, last-three-articles.php, main-nav.php, social-bar.php, and footer.php.
- main-nav.php**: The file currently open in the editor.

The content of the main-nav.php file is as follows:

```
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('/') ?&gt;"&gt;
3      &lt;img src="<?php get_theme_mod('logo') ?&gt;" class="img-fluid" alt="Logo du thème"&gt;
4    &lt;/a&gt;
5    &lt;nav class="navbar navbar-expand-md navbar-light head-menu"&gt;
6      &lt;div class="container-fluid"&gt;
7        &lt;!--
8          &lt;a class="navbar-brand" href="--&gt;<?php home_url('/') ?&gt;&lt;!--"&gt;--&gt;
9          &lt;img src="--&gt;<?php get_theme_mod('logo') ?&gt;&lt;!--" class="img-fluid" alt="Logo du thème"--&gt;
10         &lt;/a&gt;--&gt;
11         &lt;!--
12           &lt;a class="navbar-brand" href="#"--&gt;&lt;?php //var_dump(get_theme_mod('logo')) ; ?&gt;&lt;!--&gt;--&gt;
13           &lt;button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
14             &lt;span class="navbar-toggler-icon"&gt;&lt;/span&gt;
15           &lt;/button&gt;
16
17           &lt;div class="collapse navbar-collapse" id="navbarSupportedContent"&gt;
18             &lt;?php wp_nav_menu([
19               'theme_location' =&gt; 'header',
20               'container' =&gt; false,
21               'menu_class' =&gt; 'navbar-nav ml-auto mr-auto'
22             ]); ?&gt;
23             &lt;!--
24               &lt;ul class="navbar-nav mr-auto"&gt;--&gt;
25               &lt;!--
26                 &lt;li class="nav-item active"&gt;--&gt;
27                   &lt;a class="nav-link" href="#"&gt;Accueil &lt;span class="sr-only"(current)</span></a>
28                 </li>-->
29               <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 utilisé 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)

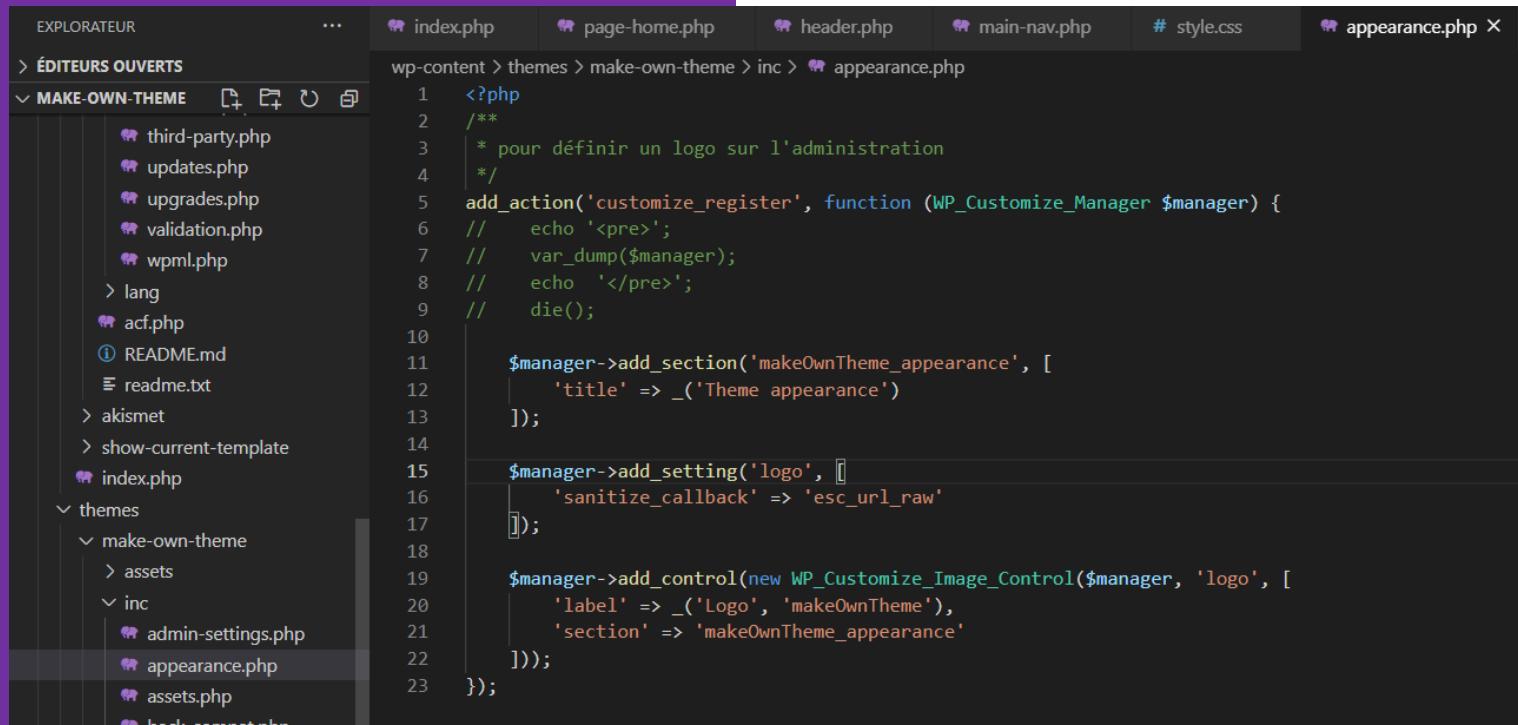
Un petit plus qui peut être appréciable pour l'utilisateur, en lui permettant d'ajouter le logo qu'il souhaite dans le backoffice.

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 » ce 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)



The screenshot shows a code editor with the following details:

- EXPLORATEUR**: Shows the directory structure of the theme, including `third-party.php`, `updates.php`, `upgrades.php`, `validation.php`, `wpml.php`, `lang`, `acf.php`, `README.md`, `readme.txt`, `akismet`, `show-current-template`, `index.php`, and the `themes/make-own-theme` folder containing `assets` and `inc` subfolders. `appearance.php` is selected in the `inc` folder.
- ÉDITEURS OUVERTS**: Shows the `index.php`, `page-home.php`, `header.php`, `main-nav.php`, `style.css`, and the currently open `appearance.php`.
- Code View**: The `appearance.php` file content is displayed:

```
<?php
/*
 * pour définir un logo sur l'administration
 */
add_action('customize_register', function (WP_Customize_Manager $manager) {
    // echo '<pre>';
    // var_dump($manager);
    // echo '</pre>';
    // die();

    $manager->add_section('makeOwnTheme_appearance', [
        'title' => __('Theme appearance')
    ]);

    $manager->add_setting('logo', [
        'sanitize_callback' => 'esc_url_raw'
    ]);

    $manager->add_control(new WP_Customize_Image_Control($manager, 'logo', [
        'label' => __('Logo', 'makeOwnTheme'),
        'section' => 'makeOwnTheme_appearance'
    ]));
});
```

Le fichier «`appearance.php` » vas 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 vas ajouter la section.

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

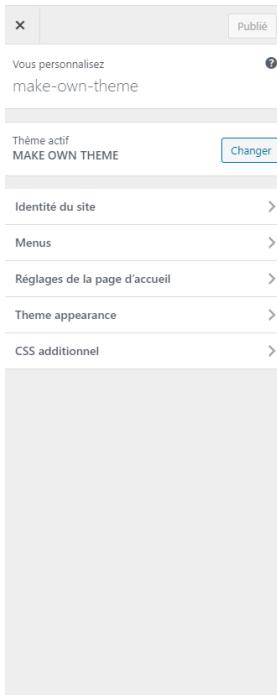
Et pour finir, l'ajout de contrôles pour l'utilisateur du backoffice.

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.



A screenshot of a website's homepage. In the top right corner, there is a header with a logo consisting of a blue mountain peak with a red flag on top, followed by the text 'Accueil' and 'A propos de nous'. Below the header, there is a section titled 'Nos dernières actualités' (Recent news) featuring three images of mountains: 'Mountain 2', 'Mountain 1', and 'Clim...'. Each news item has a small description below it.



Wordpress

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

Télécharger le walker

```
<?php
/*
 * WP Bootstrap Navwalker
 *
 * @package WP-Bootstrap-Navwalker
 *
 * @wordpress-plugin
 * Plugin Name: WP Bootstrap Navwalker
 * Plugin URI: https://github.com/wp-bootstrap/wp-bootstrap-navwalker
 * Description: A custom WordPress nav walker class to implement the Bootstrap 4 navigation style in a custom theme.
 * Author: Edward McIntyre - @twititem, WP Bootstrap, William Patton - @pattonwebz
 * Version: 4.3.0
 * Author URI: https://github.com/wp-bootstrap
 * GitHub Plugin URI: https://github.com/wp-bootstrap/wp-bootstrap-navwalker
 * GitHub Branch: master
 * License: GPL-3.0+
 * License URI: http://www.gnu.org/licenses/gpl-3.0.txt
 */
// Check if Class Exists.
if ( ! class_exists( 'WP_Bootstrap_Navwalker' ) ) {
    /**
     * WP_Bootstrap_Navwalker class.
     *
     * @extends Walker_Nav_Menu
     */
    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.

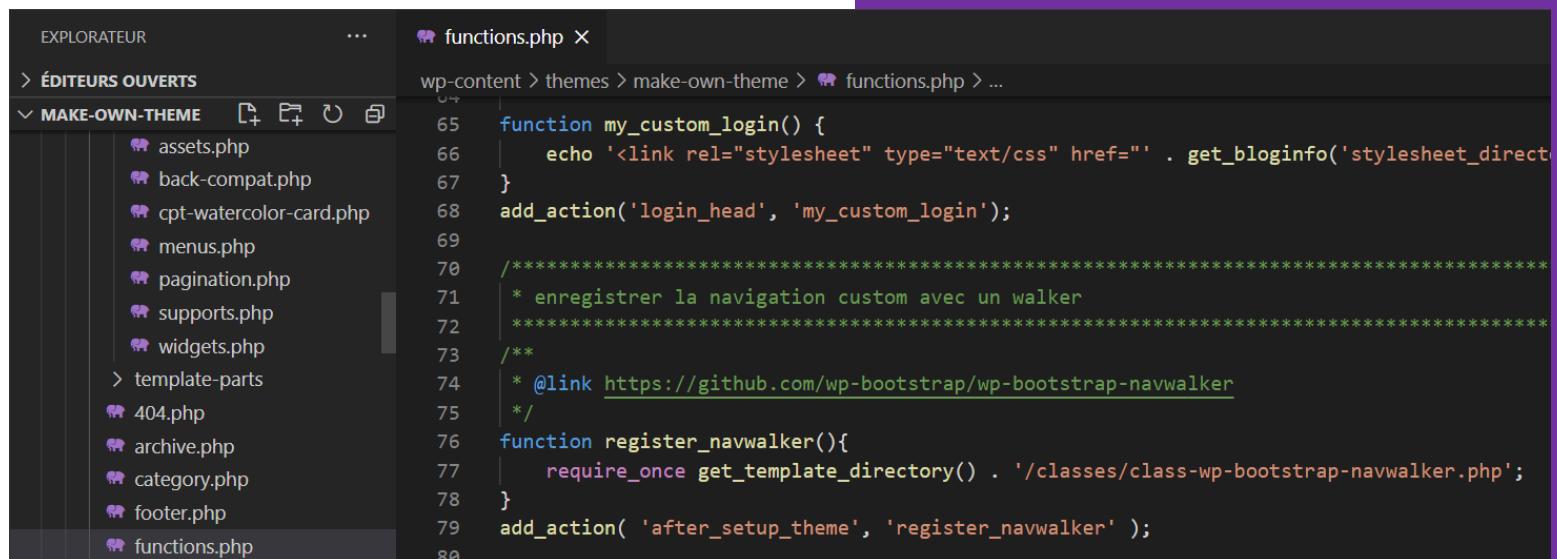
Ici, c'est le fichier: « class-wp-bootstrap-navwalker.php ».

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 corresponds à la structure ou se trouve le fichier.



The screenshot shows a code editor with the file "functions.php" open. The file is located in the "wp-content > themes > make-own-theme" directory. The code contains two main parts: a custom login function and a registration for a navigation walker. The navigation walker registration includes a comment linking to the source code on GitHub.

```
function my_custom_login() {
    echo '<link rel="stylesheet" type="text/css" href="' . get_bloginfo('stylesheet_directory') . '/classes/class-wp-bootstrap-navwalker.css' . '"';
}
add_action('login_head', 'my_custom_login');

/*
 * enregistrer la navigation custom avec un walker
 */
function register_navwalker(){
    require_once get_template_directory() . '/classes/class-wp-bootstrap-navwalker.php';
}
add_action( 'after_setup_theme', 'register_navwalker' );
```

Enregistrement des menus

The screenshot shows a code editor interface with the following details:

- EXPLORATEUR**: Shows the file structure of the theme, including `index.php`, `assets`, `classes`, `inc`, and several PHP files like `admin-settings.php`, `appearance.php`, etc.
- ÉDITEURS OUVERTS**: Shows multiple tabs: `functions.php` (active), `menus.php` (closed), `main-nav.php` (closed), `nav-menu.php` (closed), and `menu-test.php` (closed).
- Code Content (functions.php):**

```
wp-content > themes > make-own-theme > inc > menus.php > #Function#236685bb
1  <?php
2  /*
3   * ajoute le menu principal
4   */
5
6  add_action('after_setup_theme', function () {
7      //register_nav_menu('header', __('navigation principale', 'makeOwnTheme'));
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.

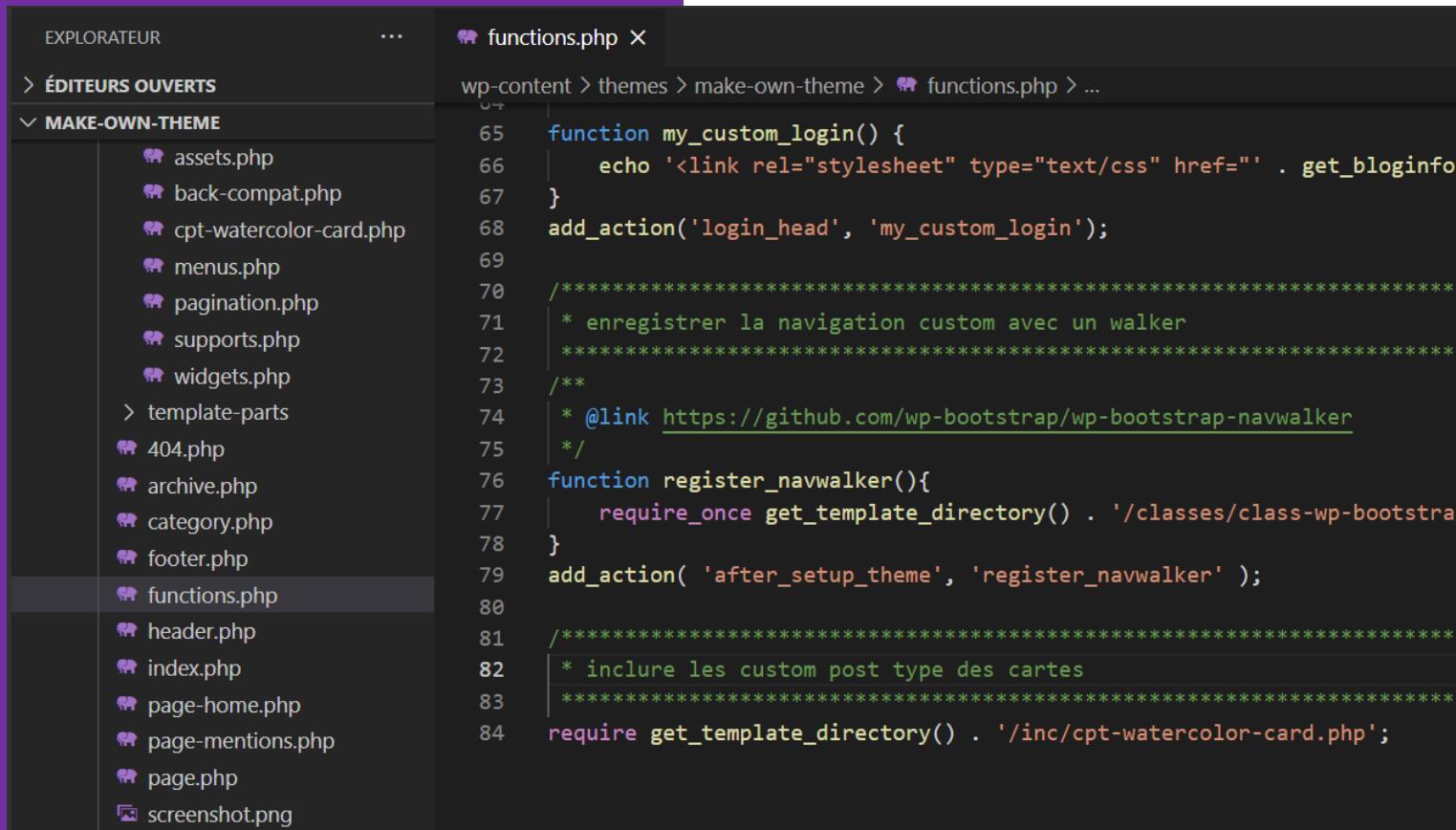
```
wp-content > themes > make-own-theme > template-parts > main-nav.php > div#head-menu.container-fluid.text-center > button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" <!-- <div class="collapse navbar-collapse" id="navbarSupportedContent" > -->
<?php
//var_dump(has_nav_menu('menu-head'));
if ( has_nav_menu('menu-head') ) {
    wp_nav_menu( array(
        // 'menu' => 'header',
        'theme_location' => 'header', // Nom que vous avez donner à l'éta
        'container' => 'div', // Elément qui contiendra votre menu, p
        'container_class' => 'navbar-collapse collapse justify-content-end'
        'container_id' => 'navbarDefault', // id du container.
        'menu_class' => 'navbar-nav', // classe de la balise ul de vot
        'menu_id' => '', // Id de la balise ul de votre menu, par
        'echo' => true, // 1 affiche le menu, 0 retourne un ob
        'fallback_cb' => 'wp_page_menu', // fonction appellée si votre m
        'before' => '', // avant la balise a du lien
        'after' => '', // après la balise a du lien
        'link_before' => '', // avant le texte de votre lien
        'link_after' => '', // après le texte de votre lien
        // 'items_wrap' => '<ul id="%1$s" class="%2$s">%3$s</ul>',
        'depth' => 2, // niveau de profondeur d'affichage. 1 n'a
        'walker' => new WP_Bootstrap_Navwalker(), // Contrôle la t
        'fallback_cb' => 'WP_Bootstrap_Navwalker::fallback', // Si vot
    ) );
} else
    echo "Veuillez assigner un menu dans l'administration Wordpress -> Appare
?>
```



Wordpress

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

Inclure les “custom post type”



The screenshot shows a code editor interface with a sidebar labeled "EXPLORATEUR" and "ÉDITEURS OUVERTS". The sidebar lists various PHP files under the "MAKE-OWN-THEME" folder. The main editor window displays the "functions.php" file, which contains the following code:

```
wp-content > themes > make-own-theme > functions.php > ...

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 * enregistrer la navigation custom avec un walker
71 ****
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 * inclure les custom post type des cartes
82 ****
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 personnalisé qui adopte le comportement des articles de Wordpress.

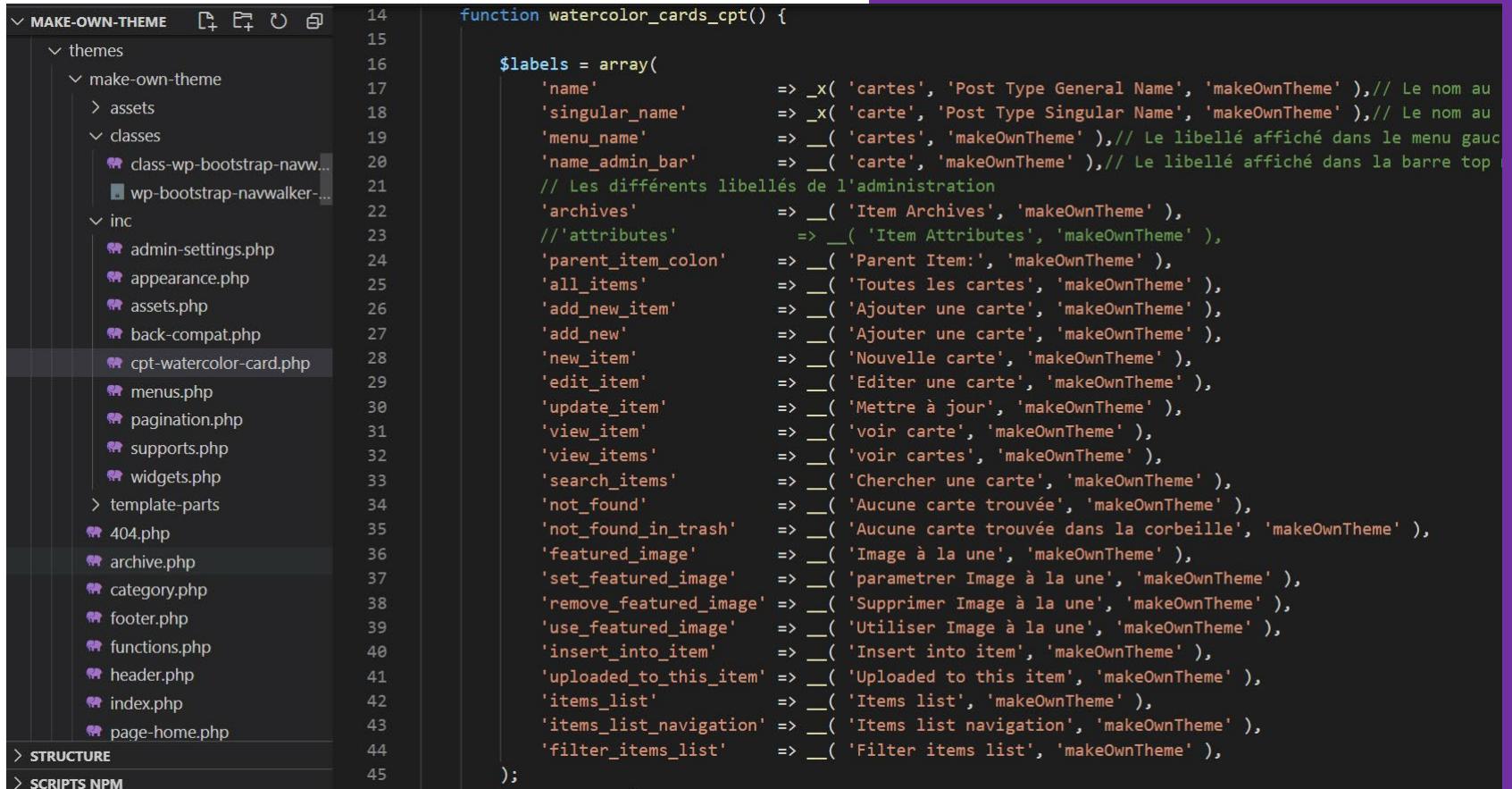
Pour commencer, il faut créer le fichier « cpt-watercolor-card » dans le dossier « inc » puis ne pas oublier de l'inclure dans le fichier « functions.php ».

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.

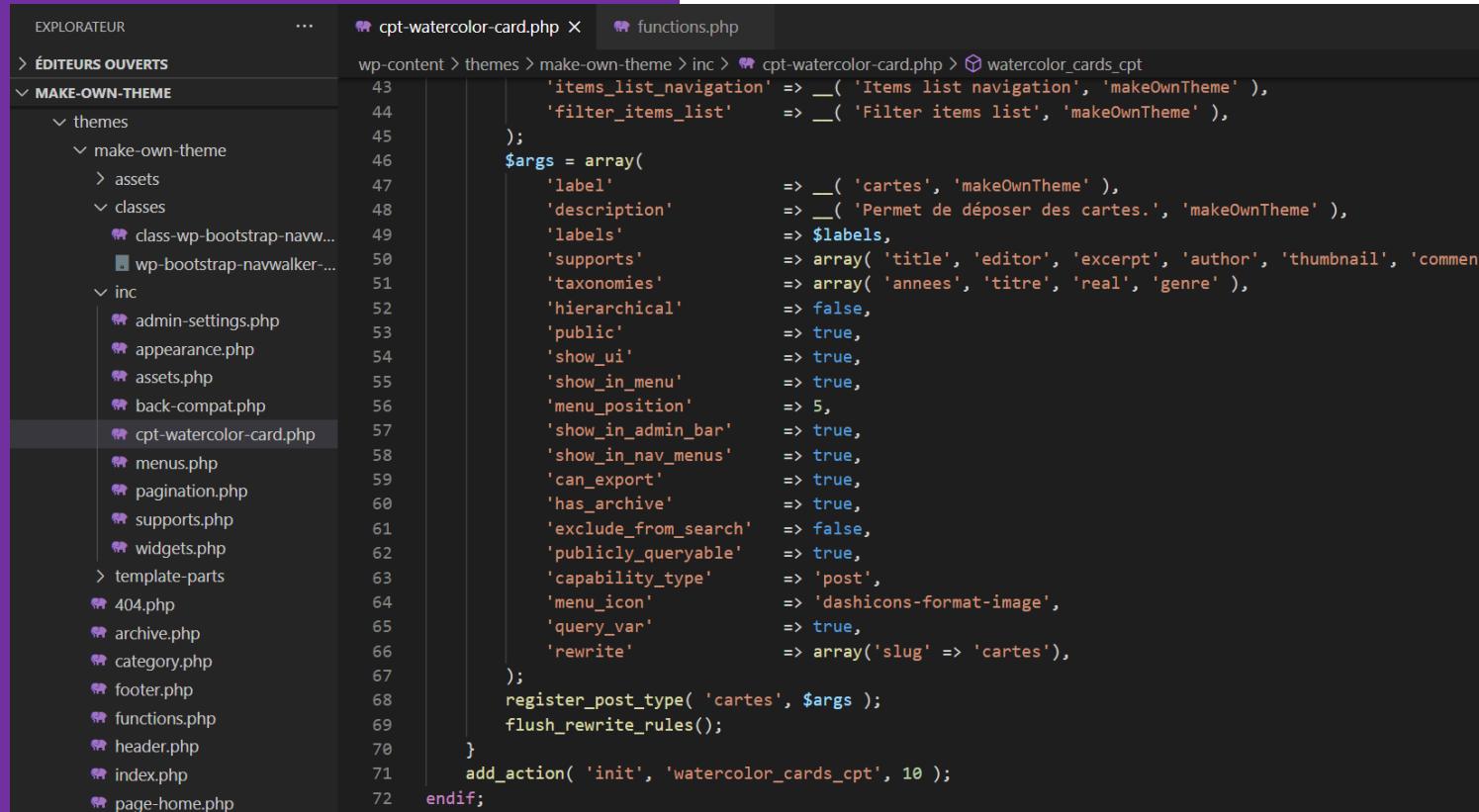
Le tableau « labels » contient tous les paramètres à mettre en place pour gérer le nom au singulier et au pluriel du « custom post type » .

Mais aussi tous les affichages des textes dans le backoffice.



```
function watercolor_cards_cpt() {
    $labels = array(
        'name'                  => __( 'cartes', 'Post Type General Name', 'makeOwnTheme' ), // Le nom au
        'singular_name'          => __( 'carte', 'Post Type Singular Name', 'makeOwnTheme' ), // Le nom au
        'menu_name'              => __( 'cartes', 'makeOwnTheme' ), // Le libellé affiché dans le menu gauche
        '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' ),
    );
}
```

Les arguments du custom post type



The screenshot shows a code editor with two tabs open: 'cpt-watercolor-card.php' and 'functions.php'. The code in 'cpt-watercolor-card.php' is as follows:

```
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;
```

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.

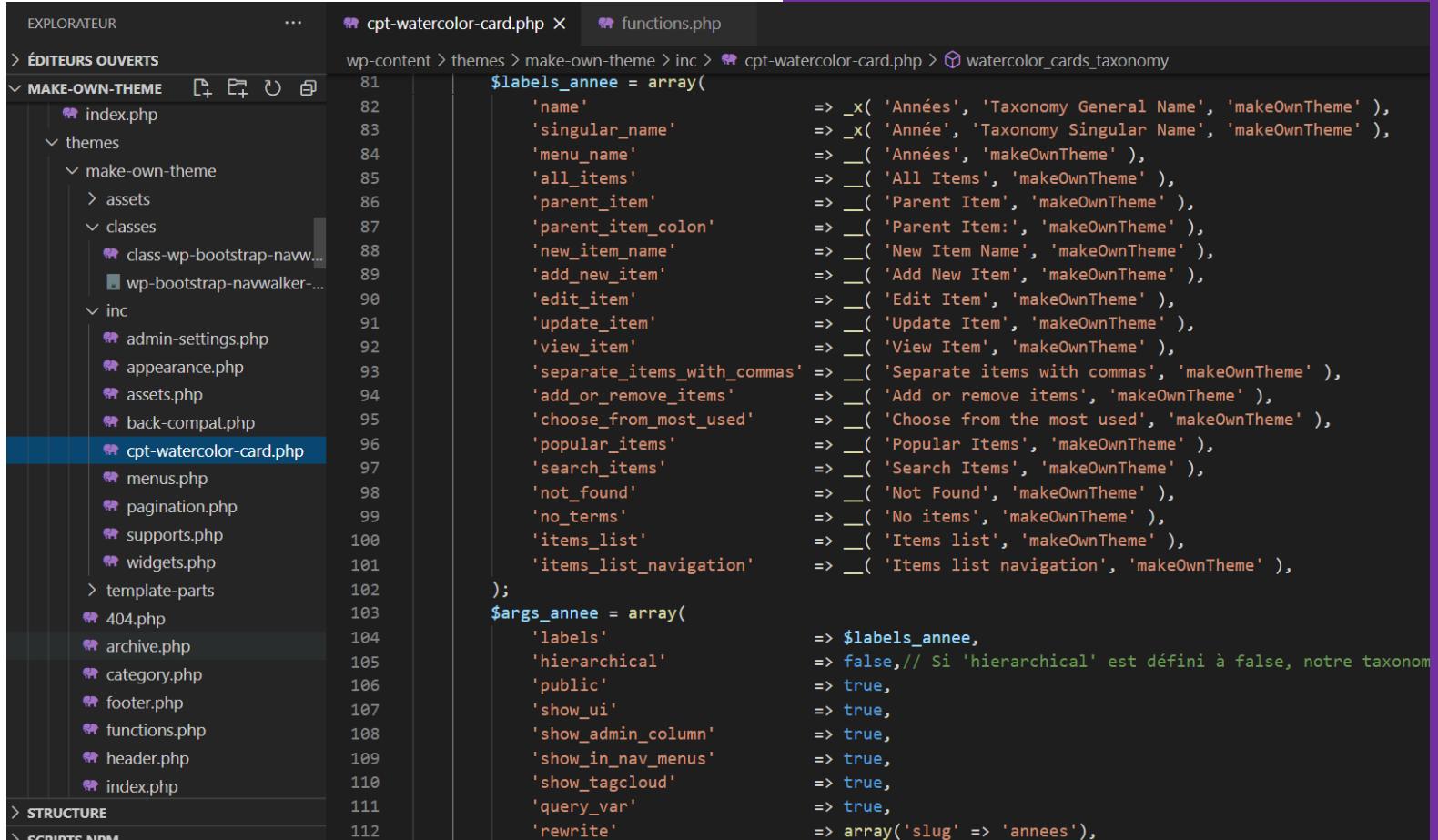
Il ne reste plus qu'à l'enregistrer grâce à la fonction « register_post_type ».

Taxonomie “année”

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

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

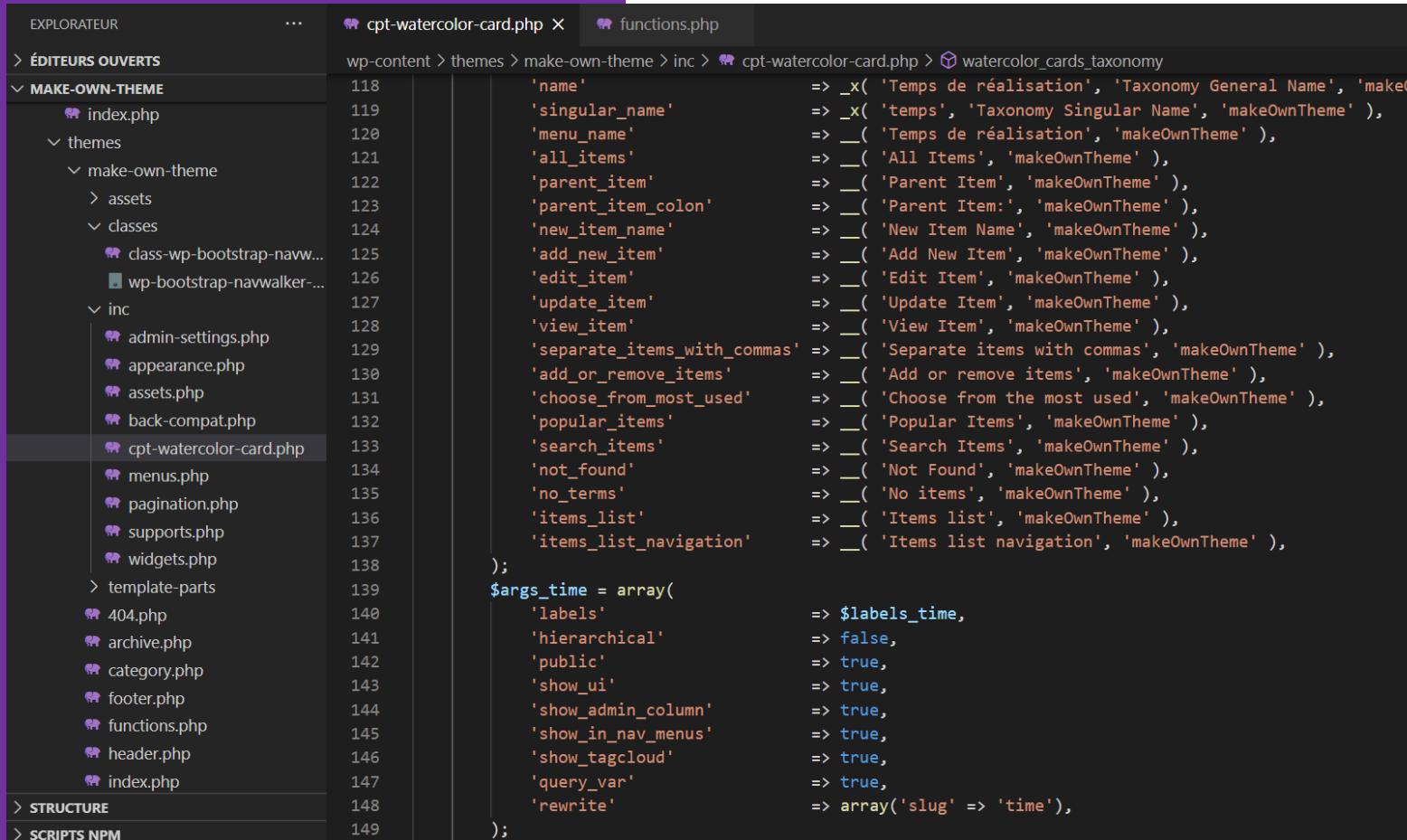
Pour la taxonomie « année », il faut déterminer les affichages pour les textes, sa hiérarchie, si son affichage est public ou non, etc...



The screenshot shows a code editor with two tabs: 'cpt-watercolor-card.php' and 'functions.php'. The 'cpt-watercolor-card.php' tab is active, displaying the following PHP code:

```
$labels_annee = array(  
    'name' => _x( 'Années', 'Taxonomy General Name', 'makeOwnTheme' ),  
    'singular_name' => _x( 'Année', 'Taxonomy Singular Name', 'makeOwnTheme' ),  
    'menu_name' => __( 'Années', 'makeOwnTheme' ),  
    'all_items' => __( 'All Items', 'makeOwnTheme' ),  
    'parent_item' => __( 'Parent Item', 'makeOwnTheme' ),  
    'parent_itemColon' => __( 'Parent Item:', 'makeOwnTheme' ),  
    'new_item_name' => __( 'New Item Name', 'makeOwnTheme' ),  
    'add_new_item' => __( 'Add New Item', 'makeOwnTheme' ),  
    'edit_item' => __( 'Edit Item', 'makeOwnTheme' ),  
    'update_item' => __( 'Update Item', 'makeOwnTheme' ),  
    'view_item' => __( 'View Item', 'makeOwnTheme' ),  
    'separate_items_with_commas' => __( 'Separate items with commas', 'makeOwnTheme' ),  
    'add_or_remove_items' => __( 'Add or remove items', 'makeOwnTheme' ),  
    'choose_from_most_used' => __( 'Choose from the most used', 'makeOwnTheme' ),  
    'popular_items' => __( 'Popular Items', 'makeOwnTheme' ),  
    'search_items' => __( 'Search Items', 'makeOwnTheme' ),  
    'not_found' => __( 'Not Found', 'makeOwnTheme' ),  
    'no_terms' => __( 'No items', 'makeOwnTheme' ),  
    'items_list' => __( 'Items list', 'makeOwnTheme' ),  
    'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),  
);  
  
$args_annee = array(  
    'labels' => $labels_annee,  
    'hierarchical' => false, // Si 'hierarchical' est défini à false, notre taxonomie  
    'public' => true,  
    'show_ui' => true,  
    'show_admin_column' => true,  
    'show_in_nav_menus' => true,  
    'show_tagcloud' => true,  
    'query_var' => true,  
    'rewrite' => array('slug' => 'annees'),  
);
```

Taxonomie “temps”



The screenshot shows a code editor with two tabs open: cpt-watercolor-card.php and functions.php. The functions.php tab is active, displaying PHP code for taxonomy declarations. The code defines a taxonomy named 'time' with various labels and arguments. The labels include 'name', 'singular_name', 'menu_name', 'all_items', 'parent_item', 'parent_item_colon', 'new_item_name', 'add_new_item', 'edit_item', 'update_item', 'view_item', 'separate_items_with_commas', 'add_or_remove_items', 'choose_from_most_used', 'popular_items', 'search_items', 'not_found', 'no_terms', 'items_list', and 'items_list_navigation'. The arguments section includes 'args_time' with labels like 'labels', 'hierarchical', 'public', 'show_ui', 'show_admin_column', 'show_in_nav_menus', 'show_tagcloud', 'query_var', and 'rewrite', along with a slug argument set to 'time'.

```
wp-content > themes > make-own-theme > inc > cpt-watercolor-card.php > watercolor_cards_taxonomy
118     'name'          => __x( 'Temps de réalisation', 'Taxonomy General Name', 'makeOwnTheme' ),
119     'singular_name' => __x( '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 « année » déclarée, c'est au tour de la taxonomie « temps ».

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

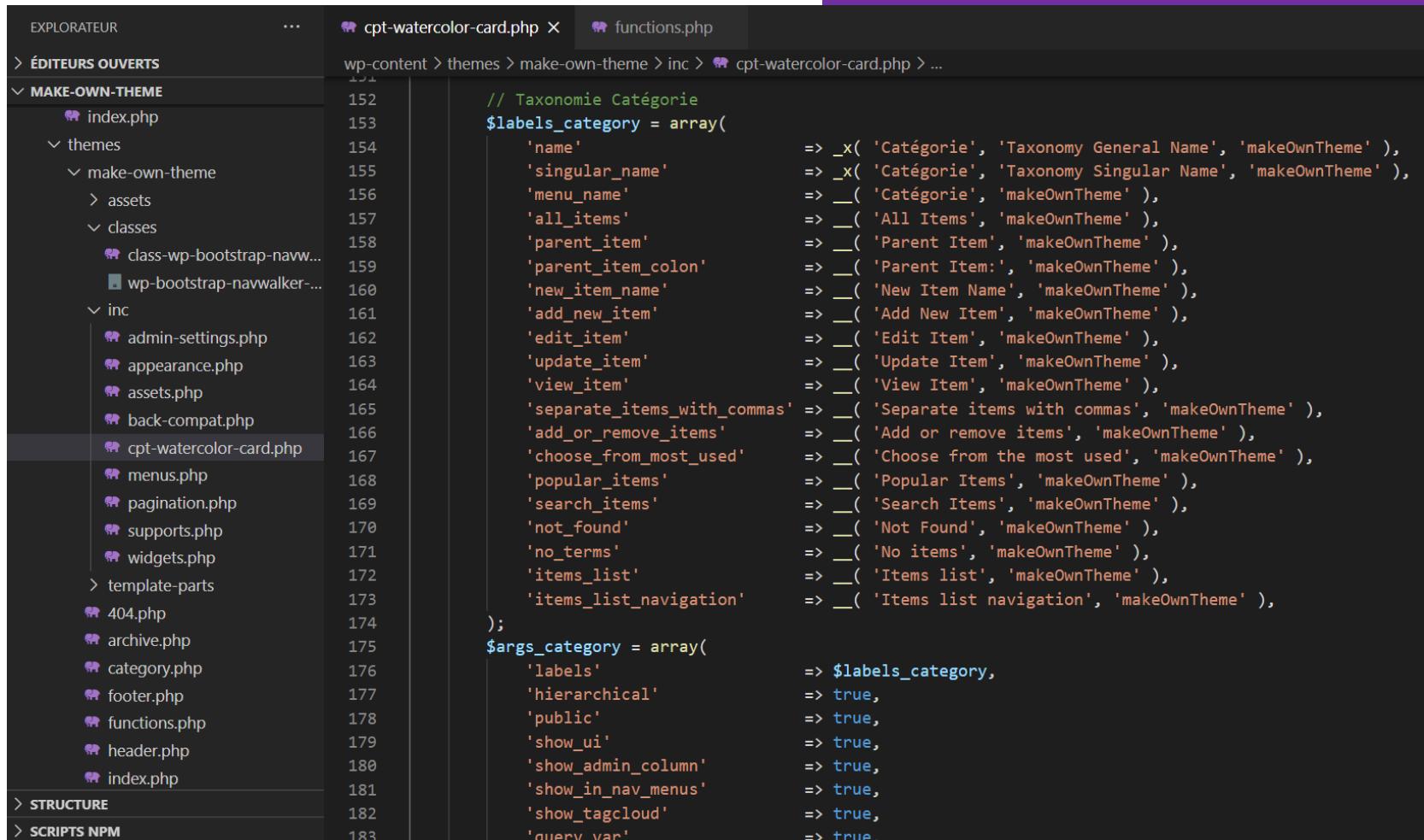
Pour la taxonomie « temps », il faut déterminer les affichages pour les textes, sa hiérarchie, si son affichage est public ou non, etc...

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 public ou non, etc...



The screenshot shows a code editor with two tabs open: 'cpt-watercolor-card.php' and 'functions.php'. The code is written in PHP and defines two arrays for a taxonomy category: '\$labels_category' and '\$args_category'.

```
// Taxonomie Catégorie
$labels_category = array(
    'name' => __( 'Catégorie', 'Taxonomy General Name', 'makeOwnTheme' ),
    'singular_name' => __( 'Catégorie', 'Taxonomy Singular Name', 'makeOwnTheme' ),
    'menu_name' => __( 'Catégorie', 'makeOwnTheme' ),
    'all_items' => __( 'All Items', 'makeOwnTheme' ),
    'parent_item' => __( 'Parent Item', 'makeOwnTheme' ),
    'parent_itemColon' => __( 'Parent Item:', 'makeOwnTheme' ),
    'new_item_name' => __( 'New Item Name', 'makeOwnTheme' ),
    'add_new_item' => __( 'Add New Item', 'makeOwnTheme' ),
    'edit_item' => __( 'Edit Item', 'makeOwnTheme' ),
    'update_item' => __( 'Update Item', 'makeOwnTheme' ),
    'view_item' => __( 'View Item', 'makeOwnTheme' ),
    'separate_items_with_commas' => __( 'Separate items with commas', 'makeOwnTheme' ),
    'add_or_remove_items' => __( 'Add or remove items', 'makeOwnTheme' ),
    'choose_from_most_used' => __( 'Choose from the most used', 'makeOwnTheme' ),
    'popular_items' => __( 'Popular Items', 'makeOwnTheme' ),
    'search_items' => __( 'Search Items', 'makeOwnTheme' ),
    'not_found' => __( 'Not Found', 'makeOwnTheme' ),
    'no_terms' => __( 'No items', 'makeOwnTheme' ),
    'items_list' => __( 'Items list', 'makeOwnTheme' ),
    'items_list_navigation' => __( 'Items list navigation', 'makeOwnTheme' ),
);
$args_category = array(
    'labels' => $labels_category,
    'hierarchical' => true,
    'public' => true,
    'show_ui' => true,
    'show_admin_column' => true,
    'show_in_nav_menus' => true,
    'show_tagcloud' => true,
    'query_var' => true,
);
```

Les “custom post type” dans le back office

The screenshot shows the WordPress back office interface. The left sidebar has a dark theme with various menu items: Tableau de bord, Articles, cartes (which is highlighted in blue), Toutes les cartes, Ajouter une carte, Années, Temps de réalisation, Catégorie, Médias, Pages, Commentaires, Apparence, Extensions, Comptes, Outils, Réglages, ACF, and Réduire le menu. The main content area is titled "cartes" and shows a list of three posts. The columns are: Titre, Auteur/autrice, Années, Temps de réalisation, Catégorie, and Date. The first post is "watercolor-3" by sylvain from 1999, 4h, animaux, published on 22/02/2021 at 17h38. The second post is "watercolor-2" by sylvain from 2002, 6h, végétations, published on 22/02/2021 at 17h37. The third post is "Watercolor-1" by sylvain from 2015, 8h, paysages, published on 22/02/2021 at 17h37. There are buttons for Ajouter une carte, Actions groupées, Appliquer, Toutes les dates, Filtrer, and Options de l'écran.

Titre	Auteur/autrice	Années	Temps de réalisation	Catégorie	Date
watercolor-3	sylvain	1999	4h	animaux	Publié 22/02/2021 à 17h38
watercolor-2	sylvain	2002	6h	végétations	Publié 22/02/2021 à 17h37
Watercolor-1	sylvain	2015	8h	paysages	Publié 22/02/2021 à 17h37

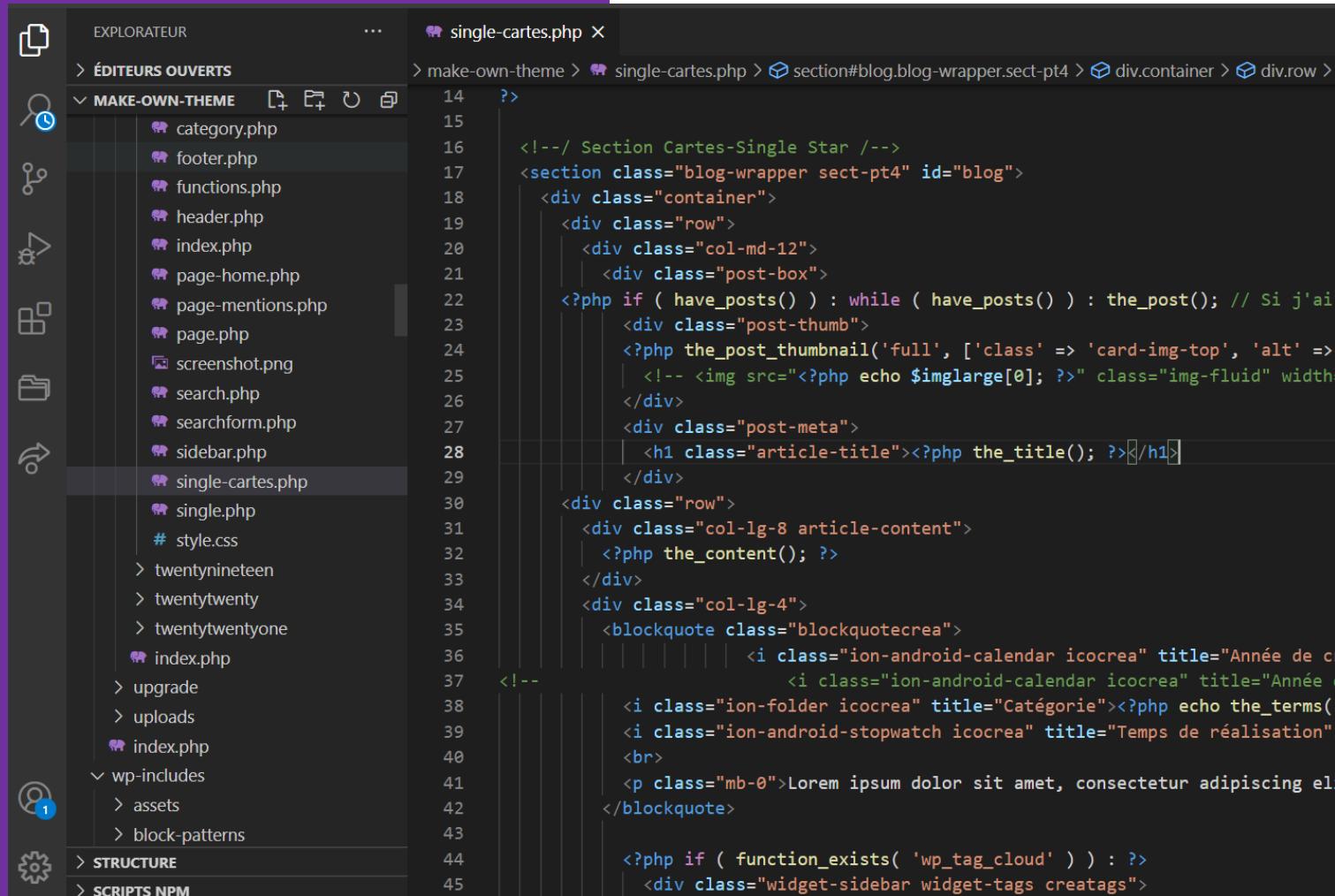
Quand le « custom post type » et ses taxonomies sont déclarés, vous devriez avoir dans votre backoffice un affichage similaire à cette capture.



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



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons for file operations like copy, paste, search, and refresh. Below the sidebar, there's a tree view of files under the 'MAKE-OWN-THEME' folder, including category.php, footer.php, functions.php, header.php, index.php, page-home.php, page-mentions.php, page.php, screenshot.png, search.php, searchform.php, sidebar.php, single-cartes.php (which is currently selected), single.php, style.css, and several twenty* files. At the bottom of the sidebar, there are sections for 'STRUCTURE' and 'SCRIPTS NPM'. The main area of the editor shows the PHP code for the 'single-cartes.php' file. The code is structured to handle posts of a specific type ('cartes-single'). It includes sections for displaying the post thumbnail, meta information (title), and content. It also includes a blockquote section with icons for calendar, folder, and stopwatch, and a placeholder for a tag cloud.

```
<?>
<!-- Section Cartes-Single Star /-->
<section class="blog-wrapper sect-pt4" id="blog">
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <div class="post-box">
                    <?php if ( have_posts() ) : while ( have_posts() ) : the_post(); // Si j'ai
                        <div class="post-thumb">
                            <?php the_post_thumbnail('full', ['class' => 'card-img-top', 'alt' =>
                                | <!-- 
                            </div>
                            <div class="post-meta">
                                | <h1 class="article-title"><?php the_title(); ?></h1>
                            </div>
                        <div class="row">
                            <div class="col-lg-8 article-content">
                                | <?php the_content(); ?>
                            </div>
                            <div class="col-lg-4">
                                <blockquote class="blockquotecrea">
                                    | | | | | <i class="ion-android-calendar icocrea" title="Année de cr">
                                    | | | | | <i class="ion-android-calendar icocrea" title="Année de cr">
                                    | | | | | <i class="ion-folder icocrea" title="Catégorie"><?php echo the_terms(>
                                    | | | | | <i class="ion-android-stopwatch icocrea" title="Temps de réalisation">
                                    | | | | | <br>
                                    | | | | | <p class="mb-0">Lorem ipsum dolor sit amet, consectetur adipiscing eli
                                </blockquote>
                            </div>
                        <!--
                            <?php if ( function_exists( 'wp_tag_cloud' ) ) : ?>
                            <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 ce nommage « single-cpt.php ».

Le terme « cpt » sera remplacé 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 interface with a dark theme. On the left is a sidebar titled "EXPLORATEUR" containing a tree view of files and folders. The main area shows three tabs at the top: "single-cartes.php", "taxonomy-annees.php", and "taxonomy-categorie.php". The "taxonomy-annees.php" tab is active. The code editor displays the content of the "taxonomy-annees.php" file, which is a PHP template for displaying custom post type taxonomy pages. The code includes comments explaining the template hierarchy and package information. It uses the "get_header()" function and includes logic for handling attachments and filters.

```
<?php
/*
 * The template for displaying CPT creations taxonomy categorie
 *
 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
 *
 * @package WordPress
 * @subpackage make own theme
 * @since 1.0.0
 */
get_header();

//on stocke l'url de l'image à la une
$imglarge = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ),

//on récupère l'attribut alt lié à l'image
$alttext = get_post_meta ( get_post_thumbnail_id( $post->ID ), '_wp_attachment_metadata' );
?>

<!-- Section Taxonomy annee Begin /--&gt;
&lt;section id="portfolio" class="portfolio-mf sect-pt4"&gt;
    &lt;div class="container"&gt;
        &lt;div class="row"&gt;
            &lt;div class="col-sm-12"&gt;
                &lt;div class="title-box"&gt;
                    &lt;div id="filters" class="container-fluid mb-3 mt-5"&gt;
                        &lt;?php
                            // Affiche toutes les catégories avec liens
                            $terms = get_terms( 'annees', array(
                                'order'      =&gt; 'ASC',
                                'hide_empty' =&gt; 0,
                                'parent'     =&gt; 0,
                                'orderby'    =&gt; 'name',
                                'limit'      =&gt; 100
                            ) );
                            foreach ( $terms as $term ) {
                                echo '&lt;a href="#"&gt;' . $term-&gt;name . '&lt;/a&gt; ';
                            }
                        &lt;/?php
                    &lt;/div&gt;
                &lt;/div&gt;
            &lt;/div&gt;
        &lt;/div&gt;
    &lt;/div&gt;
&lt;/section&gt;</pre>
```

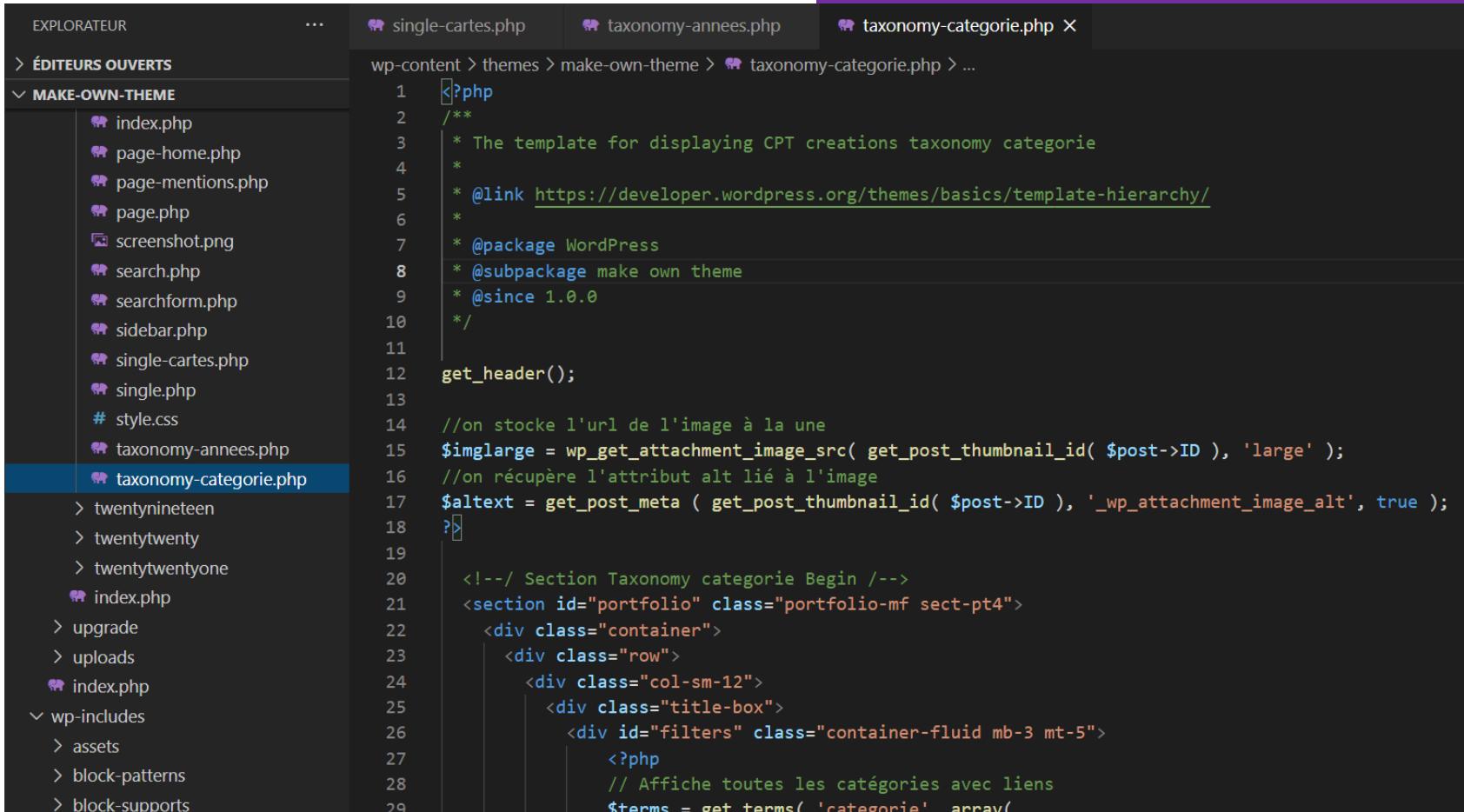
En mettant en place le fichier « taxonomy-annees.php », il est possible de gérer une page qui affichera une liste de « custom post type » qui ont une taxonomie « années » en particulier.

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 WordPress theme editor interface. On the left, there's an 'EXPLORATEUR' sidebar listing files in the 'MAKE-OWN-THEME' directory, including index.php, page-home.php, page-mentions.php, page.php, screenshot.png, search.php, searchform.php, sidebar.php, single-cartes.php, single.php, style.css, taxonomy-annees.php, and taxonomy-categorie.php. The 'taxonomy-categorie.php' file is currently selected and highlighted in blue. The main area displays the PHP code for this template. The code includes comments explaining it's for displaying CPT creations taxonomy categorie, links to developer documentation, and package information. It starts with a standard header and then retrieves the large image URL for the post thumbnail. Below that, it begins a section for taxonomy categories, creating a container and a row with a column for titles. A filters section is also present, which includes a call to get_terms() with the parameter 'categorie'.

```
<?php
/*
 * The template for displaying CPT creations taxonomy categorie
 *
 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
 *
 * @package WordPress
 * @subpackage make_own_theme
 * @since 1.0.0
 */
get_header();

//on stocke l'url de l'image à la une
$imglarge = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'large' );
//on récupère l'attribut alt lié à l'image
$alttext = get_post_meta( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
?>

<!-- Section Taxonomy categorie Begin -->
<section id="portfolio" class="portfolio-mf sect-pt4">
    <div class="container">
        <div class="row">
            <div class="col-sm-12">
                <div class="title-box">
                    <div id="filters" class="container-fluid mb-3 mt-5">
                        <?php
                            // Affiche toutes les catégories avec liens
                            $terms = get_terms( 'categorie', array(

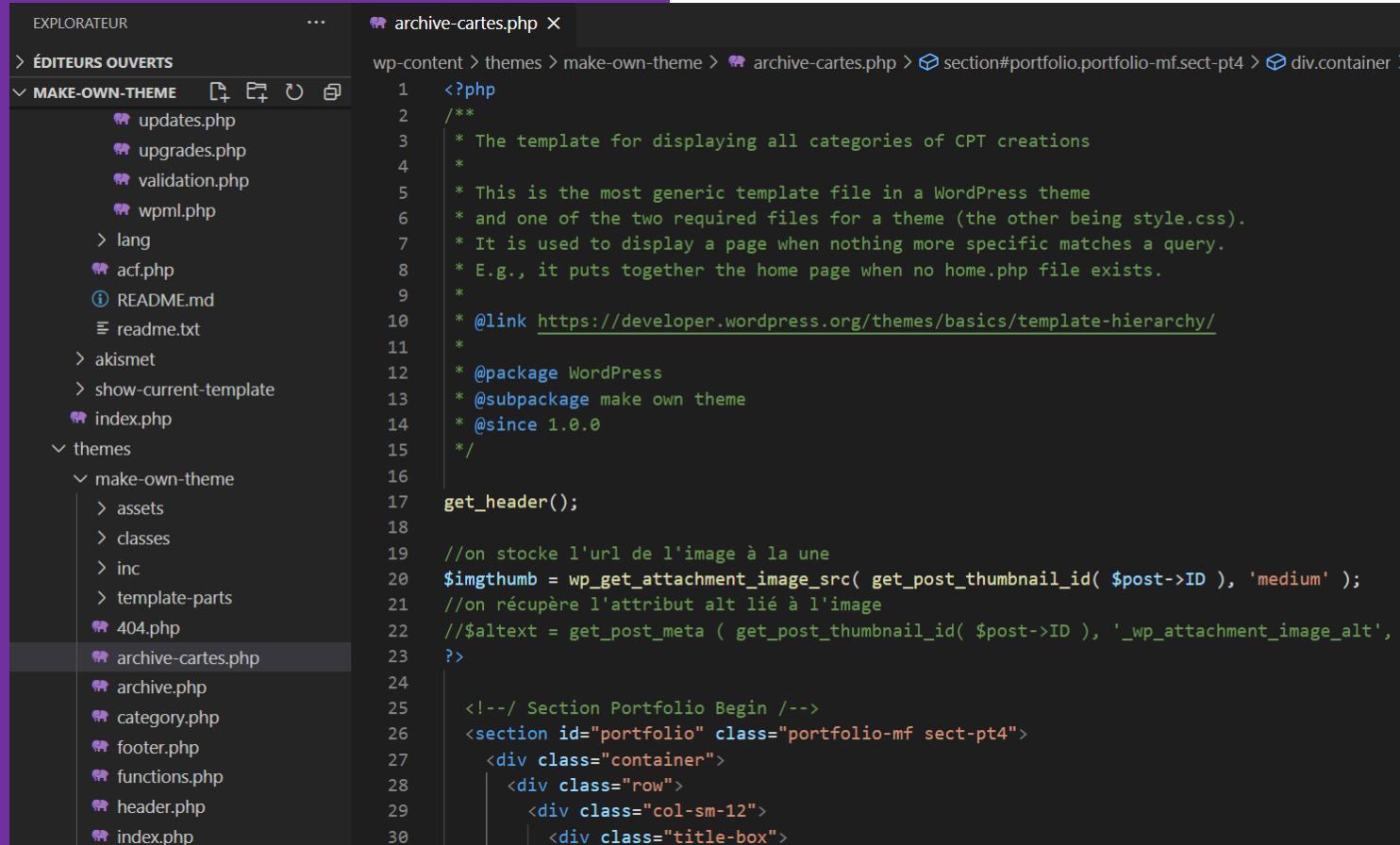
```



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 interface with a dark theme. On the left, there's a sidebar titled "EXPLORATEUR" and "ÉDITEURS OUVERTS". Under "ÉDITEURS OUVERTS", there's a section for "MAKE-OWN-THEME" containing files like "updates.php", "upgrades.php", "validation.php", "wpml.php", "lang", "acf.php", "README.md", and "readme.txt". Below that, there are sections for "akismet", "show-current-template", "index.php", and "themes/make-own-theme" which includes "assets", "classes", "inc", "template-parts", "404.php", "archive-cartes.php" (which is currently selected), "archive.php", "category.php", "footer.php", "functions.php", "header.php", and "index.php". The main area of the editor shows the content of the "archive-cartes.php" file. The code is written in PHP and includes comments explaining its purpose, such as displaying all categories of CPT creations and handling generic template logic. It also includes metadata for the WordPress package and subpackage.

```
<?php
/*
 * The template for displaying all categories of CPT creations
 *
 * This is the most generic template file in a WordPress theme
 * and one of the two required files for a theme (the other being style.css).
 * It is used to display a page when nothing more specific matches a query.
 * E.g., it puts together the home page when no home.php file exists.
 *
 * @link https://developer.wordpress.org/themes/basics/template-hierarchy/
 *
 * @package WordPress
 * @subpackage make own theme
 * @since 1.0.0
 */
get_header();

//on stocke l'url de l'image à la une
$imgthumb = wp_get_attachment_image_src( get_post_thumbnail_id( $post->ID ), 'medium' );
//on récupère l'attribut alt lié à l'image
//$alttext = get_post_meta( get_post_thumbnail_id( $post->ID ), '_wp_attachment_image_alt', true );
?>

<!-- Section Portfolio Begin /-->
<section id="portfolio" class="portfolio-mf sect-pt4">
    <div class="container">
        <div class="row">
            <div class="col-sm-12">
                <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 appellée 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 : Créer Modèles existants :

Méthode d'exportation :

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

Format :

Tables :

	Tables	Structure	Données
<input type="checkbox"/>	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.

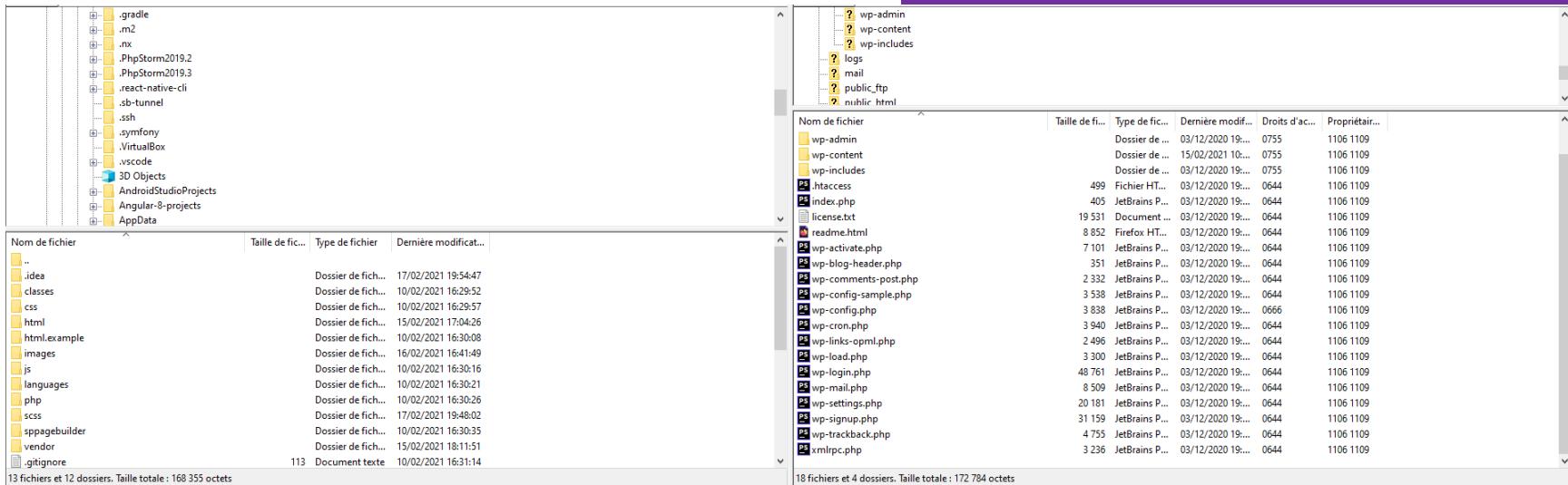
Il vous suffit ensuite de cliquer sur « exporter » pour avoir accès à l'exportation des tables depuis la base de données concernée.

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 'search/replace' interface of the srdb tool. At the top, it says 'replace http://www.ancienneurl.' with 'http://www.nouvelleurl.' with an unchecked 'use regex' option. Below this, under 'database', there are fields for 'name' (redacted), 'user' (root), 'pass' (root), and 'host' (localhost). Under 'tables', there are radio buttons for 'all tables' (selected) and 'select tables', and input fields for 'columns to exclude' (eg. guid) and 'columns to include only' (eg. post_content, post_excerpt). Under 'actions', there are red buttons for 'update details', 'dry run', 'live run', and two more options: 'convert to innodb' and 'convert to utf8 unicode'. At the bottom, under 'delete', there is a red button labeled 'delete me' 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.