

IUT d'Aix-Marseille site d'Arles

BUT Informatique

**Robyn Tambon**

# **Stage en environnement professionnel**

## **Création du site institutionnel de la communauté d'agglomération Arles Crau Camargue Montagnette**

15 semaines en Entreprise

Organisme d'accueil :  
Communauté d'agglomération  
ACCM

Département informatique  
site d'Arles

Maître de stage :  
Lilia Bekrar

Tuteur :  
B. Colombel

# Table des matières

Remerciement.....	4
Introduction .....	5
Présentation de l'entreprise .....	6
La collectivité : la Communauté d'agglomération ACCM (Arles, Crau, Camargue et Montagnette) et son positionnement .....	6
Historique.....	6
Priorités de modernisation .....	7
Organisation interne, cellule d'accueil et Moyens informatiques .....	7
Organigramme .....	9
Sujet du stage .....	10
Présentation du travail accompli .....	11
Appropriation de WordPress et de son écosystème.....	11
Outils de gestion de projet et mise en place des outils front-end.....	11
Premiers pas vers l'intégration visuelle .....	12
Intégration des maquettes et découverte des outils avancés .....	14
Structuration des contenus et premières logiques dynamiques .....	16
Restructuration des contenus et personnalisation de l'éditeur Gutenberg.....	19
Étapes de la création de bloc custom .....	21
Génération du squelette du bloc via WP-CLI .....	21
Mise en place de Babel.....	22
Configuration de Webpack .....	23
Dynamisme des blocs : pourquoi ce choix ?.....	23
Structure du fichier JSX .....	24
Enregistrement et rendu côté PHP.....	25
Gestion des pictogrammes et autres champs des custom-categories .....	30
Approfondissement des templates et stabilisation des blocs personnalisés .....	33
Amélioration du template des taxonomies.....	33
Finalisation des pages principales et navigation.....	34
Création de la page Actualités .....	34
Consolidation éditoriale et montée en flexibilité des blocs.....	35
Structuration progressive des contenus.....	35
Affichage dynamique des actualités .....	36
Ajustements sur les blocs Gutenberg personnalisés .....	37
Améliorations de la page d'accueil.....	37
Avancement sur les templates secondaires .....	38
Vie d'équipe et observations humaines .....	38

Renforcement de la navigation et finalisation des composants dynamiques .....	39
Affinage des templates de recherche et d'erreur.....	39
Évolutions sur les blocs Gutenberg .....	39
Implémentation d'un système de menu dynamique.....	39
Revue d'étape et retours .....	41
Améliorations diverses.....	42
Blocs dynamiques via la page de menu "Réglage Accueil" .....	43
Difficultés techniques rencontrées.....	44
Vers une meilleure maintenabilité : de TailwindCSS à SASS.....	45
Abandon de TailwindCSS.....	45
Retours utilisateurs et implémentation .....	45
Gestion d'un bug ACF complexe .....	46
Finalisation de la suppression de Tailwind .....	46
Décision institutionnelle et report du projet.....	51
Améliorations du back-office, tri personnalisé et lancement de la rédaction.....	51
Mise en place de la pagination native WordPress.....	52
Développement d'une fonctionnalité de tri personnalisée pour les pages sommaires .....	52
Amélioration de l'expérience utilisateur en back-office .....	53
Résolution d'un bug dans "Réglage Accueil" .....	53
Mise en pause officielle du projet côté service Application .....	53
Conclusion.....	54
Résumé .....	55
Mots-clés .....	55
Annexes techniques.....	56
Structure des templates WordPress .....	56
Arborescence du thème WordPress ACCM .....	57
Feuille de temps .....	58
Glossaire.....	59
Lexique.....	60
Bibliographie.....	63
Ressources textuelles .....	63
Ressources vidéo.....	63

## Remerciement

Avant tout, je tiens à exprimer ma profonde gratitude à Lilia Bekrar, ma tutrice de stage, pour sa bienveillance, sa disponibilité et sa gentillesse pendant ces semaines passées au sein de la Mairie d'Arles. Elle a su m'accueillir dès le début dans une bonne ambiance et m'a accompagné dans la découverte du fonctionnement du service. Toujours à l'écoute et encourageante, elle a fait de ce stage une expérience motivante.

Je remercie également Frederic Cases, le développeur expérimenté du service. Il a su m'encadrer avec rigueur. Grâce à lui, j'ai été guidé tout au long de ce stage dans un cadre de travail adapté aux attentes des utilisateurs et par son expérience, j'ai mieux compris les attentes du métier de développeur web.

Je souhaite aussi remercier Pierre Billes, pour sa présence, sa bonne humeur communicative et les échanges toujours intéressants que nous avons pu avoir. Sa personnalité chaleureuse a apporté une dynamique conviviale au quotidien, et m'a permis de me sentir pleinement intégré au sein de l'équipe.

Je remercie également Samia Rabia, pour sa présence au sein du service et les échanges cordiaux que nous avons pu avoir.

Mes remerciements vont aussi à Didier Marion, chef du service applications, qui, à travers son expertise et ses interventions lors des réunions, m'a permis de mieux cerner les enjeux techniques liés aux projets du service. Son regard professionnel a apporté un éclairage essentiel à ma compréhension globale du fonctionnement des systèmes d'information.

Je tiens tout particulièrement à remercier Nicolas Issart, chef de la Direction des Systèmes d'Information, sans qui ce stage n'aurait tout simplement pas été possible. Je lui suis sincèrement reconnaissant de m'avoir offert cette opportunité.

Enfin, je remercie l'ensemble des agents de la Direction des Systèmes d'Information pour leur accueil, leur gentillesse et leur collaboration ainsi qu'Éléonore Dherbecourt, designeuse du projet, pour ses conseils avisés et sa disponibilité, notamment en matière de bonnes pratiques CSS et de respect de la cohérence graphique. Travailler à leurs côtés a été une chance, et je garderai de ce stage un souvenir à la fois professionnel et humain très positif. Bien sûr, mes remerciements vont aussi à l'IUT et Mr B. Colombel, mon tuteur de l'IUT.

## Introduction

Dans le cadre de ma troisième année de formation en informatique, j'ai eu la chance de faire un stage de 15 semaines, du 17 février au 30 mai 2025, à la DSI (Direction des Systèmes d'Information) de la Mairie d'Arles. J'ai été accueilli au sein du service Applications, où l'on m'a confié un vrai projet concret qui a été un gros chantier, mais aussi une belle opportunité pour apprendre.

Ce projet visait une refonte du site institutionnel de la communauté d'agglomération ACCM (Arles Crau Camargue Montagnette). L'ancien site, basé sur le CMS MODX, présentait des limites : il n'était plus adapté aux usages actuels, que ce soit au niveau du design, de la compatibilité mobile ou de la facilité de gestion. Il fallait repartir de zéro avec quelque chose de plus moderne, évolutif et surtout simple à administrer. C'est là que j'ai pu intervenir, en participant à la création de ce nouveau site WordPress.

Très vite, je me suis retrouvé face à une problématique qu'on rencontre quelquefois dans une administration : **comment concevoir un site WordPress à la fois moderne, responsive, respectant une maquette graphique, tout en restant accessible et facilement administrable par des agents maîtrisant peu la technique.** Cette question m'a accompagnée tout au long de mon stage, et beaucoup influencé mes choix techniques.

Afin de relever ce défi, j'ai appris à utiliser de nouveaux outils, à travailler avec différents profils (développeurs, agents, graphiste...), et surtout à organiser mon travail en appliquant de bonnes pratiques de développement. Cela a été pour moi l'occasion de progresser en PHP, en création de thèmes WordPress, en intégration web, mais aussi en gestion de projet. Cette seconde expérience dans une collectivité, m'a permis de mieux comprendre le fonctionnement d'un service informatique d'une mairie.

Ce mémoire revient sur cette expérience. Il est structuré en trois parties : pour commencer, une présentation de l'environnement de travail et du contexte institutionnel, ensuite le descriptif des missions que j'ai menées, surtout sur le plan technique, et enfin une analyse plus personnelle sur ce que m'a enseigné ce stage, les points positifs comme les difficultés rencontrées.

## Présentation de l'entreprise

### La collectivité : la Communauté d'agglomération ACCM (Arles, Crau, Camargue et Montagnette) et son positionnement

La Communauté d'agglomération ACCM est un établissement public de coopération intercommunale créé en 2004. Située dans le département des Bouches-du-Rhône, en région Provence-Alpes-Côte d'Azur, elle regroupe six communes : Arles, Saint-Martin-de-Crau, Tarascon, Saintes-Maries-de-la-Mer, Boulbon et Saint-Pierre-de-Mézoargues. Le territoire de l'ACCM s'étend sur 1 445,8 km<sup>2</sup> et comptait en 2021, 83 429 habitants .

Ce vaste territoire présente une diversité géographique notable, incluant des zones urbaines, rurales, ainsi que des espaces naturels protégés tels que la Camargue et la Crau. Le siège de l'ACCM est situé au Parc des Ateliers, 5 rue Yvan Audouard, 13200 Arles.

Le conseil communautaire réunit des élus issus des différentes communes membres, qui participent ensemble aux décisions importantes pour le territoire. Actuellement, la présidence de l'ACCM est assurée par Monsieur Patrick de Carolis, également maire de la Ville d'Arles. La communauté d'agglomération exerce plusieurs compétences essentielles pour le développement et la gestion du territoire, notamment :

- Le développement économique et l'aménagement de l'espace communautaire.
- La politique de l'habitat et la politique de la ville.
- La gestion de l'eau et de l'assainissement.
- La gestion des déchets ménagers et assimilés.
- La construction, l'aménagement, l'entretien et la gestion des équipements culturels et sportifs d'intérêt communautaire.

### Historique

La Communauté d'agglomération Arles Crau Camargue Montagnette est engagée dans plusieurs projets visant à renforcer l'attractivité et la durabilité de son territoire.

Elle accompagne par exemple les entreprises locales dans leurs investissements, en partenariat avec la région Sud, via différents dispositifs d'aides. De plus, elle accorde une attention particulière aux enjeux environnementaux, en travaillant sur la prévention des risques naturels tels que les inondations ou les incendies.

Dans le prolongement de ces actions, L'ACCM mutualise certains moyens pour permettre de rationaliser son action et d'assurer une continuité de service sur l'ensemble du territoire, notamment le service informatique.

C'est dans ce contexte que la Direction des Systèmes d'Information de la Ville d'Arles a été mutualisée avec l'ACCM.

**Bien que mon organisme d'accueil soit officiellement la communauté d'agglomération ACCM, mon intégration s'est faite au sein de la DSI de la mairie d'Arles.**

### Priorités de modernisation

Ainsi, Depuis 2002, la Ville d'Arles s'était engagée dans le développement de **logiciels libres** pour ses besoins métiers internes. Ce choix stratégique a permis, dès 2011, de générer des économies estimées à 520 000 euros, selon un article de La Gazette.

Bien que ces données soient anciennes, elles illustrent l'engagement de la collectivité en faveur de solutions open-source.

Les investissements de la Ville portent également sur la formation continue des agents, l'amélioration des équipements, et la modernisation des services publics.

Cette politique se traduit concrètement par la mise en place de projets numériques, comme la refonte complète du site institutionnel de l'ACCM.

### Organisation interne, cellule d'accueil et Moyens informatiques

La mairie est structurée selon une hiérarchie composée de directions générales, de directions fonctionnelles et de services spécialisés.

La coordination est assurée par la Direction Générale des Services, qui supervise les différentes directions générales (espaces publics, attractivité du Territoire, ressources...).

Parmi ces directions, la DSI joue un rôle central dans la transformation numérique de la collectivité. Cette direction est caractérisée par :

- 2 Pôles
- 3 Services

Et doit gérer :

- 1749 Utilisateurs
- 1400 Ordinateurs
- 127 Sites Distants
- 143 Copieurs Multifonction
- 140 Applications
- 217 Serveurs Virtuels
- 350 Objets Connectés LoRa
- 350 Tablettes
- 750 Smartphones
- 330 Clients Entreprises ACCM THD

La DSI regroupe environ 20 agents répartis en services et pôles :

Service Applications, Service Infrastructures, Service Réseaux Télécommunications THD, Pôle Support, Pôle Gestion Administratif.

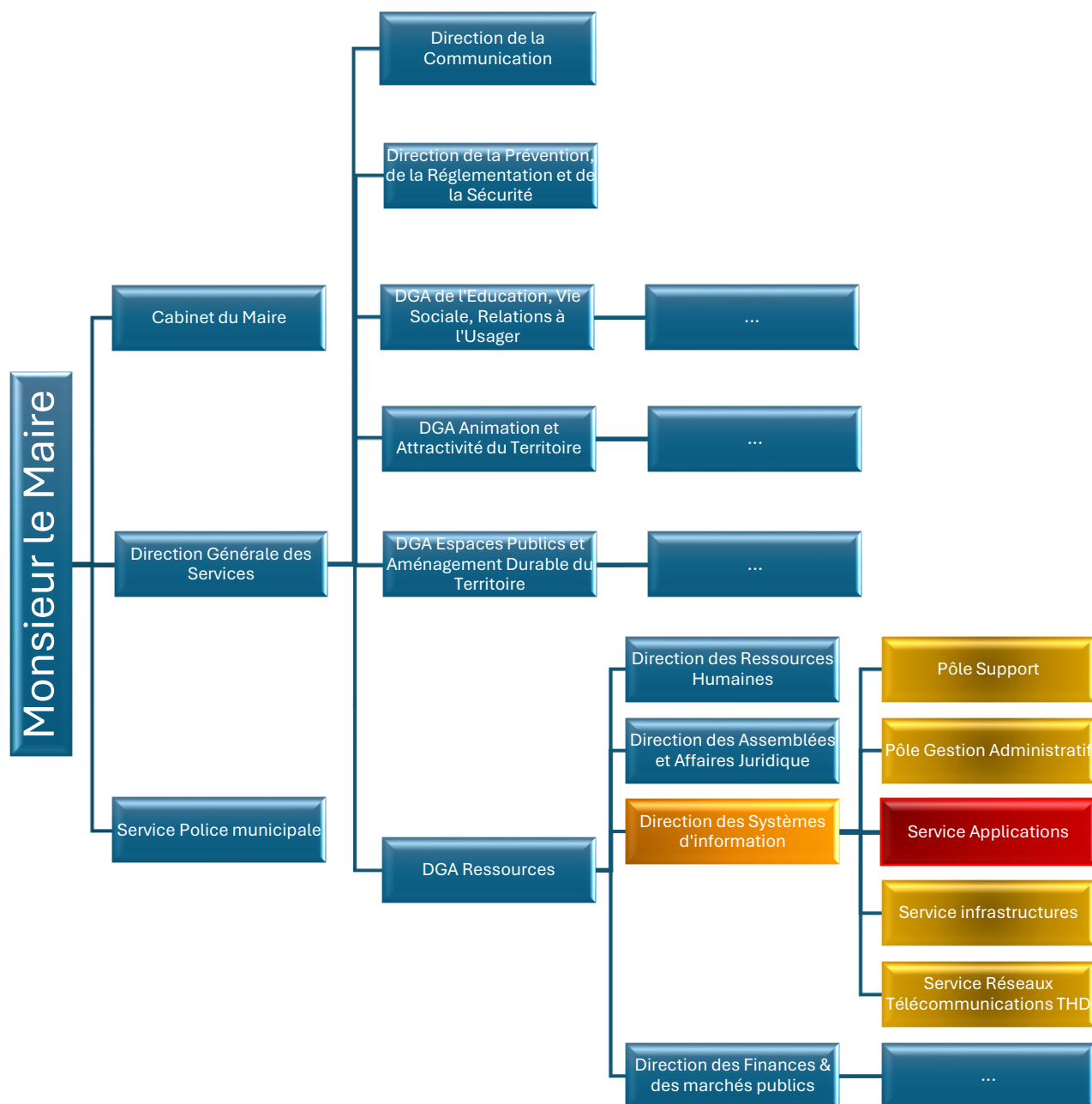
C'est au sein du service Applications que j'ai effectué mon stage.

Le service Applications accompagne les services fonctionnels à la mise en place et l'utilisation d'outils numériques.

A ce titre, il veille à la bonne intégration des nouveaux logiciels dans l'architecture technique du système d'information, en s'appuyant sur les autres services de la DSI.

Il se doit également de travailler en étroite collaboration avec les services fonctionnels.

## Organigramme



## Sujet du stage

Le stage que j'ai effectué au sein de la Direction des Systèmes d'Information de la Mairie d'Arles avait pour objectif : **la refonte complète du site web de l'ACCM**.

Ce projet m'a été confié dès mon arrivé au service Applications, avec pour mission de concevoir et de développer ce nouveau site sous **WordPress**, en respectant une charte graphique.

L'ancien site avait été réalisé avec le **CMS MODX**, et souffrait d'obsolescence technique et graphique. La collectivité a donc fait le choix de repartir de zéro, en utilisant un outil plus actuel, **WordPress**.

Ce développement devait respecter la charte graphique conçue par la designeuse Éléonore Dherbecourt, amenant une navigation fluide, une gestion de contenu fonctionnelle, et une compatibilité élargie.

Mon stage m'a amené à :

- Installer et configurer un environnement local de développement.
- Créer un thème WordPress sur mesure.
- Mettre en place une architecture de contenus solide (pages, taxonomies personnalisées, liaison pictogrammes, Custom Post Type, page de menu, etc.).
- Développer des blocs Gutenberg personnalisés.
- Gérer l'interface frontend du site.
- Mettre en place le responsive design.

## Présentation du travail accompli

### Appropriation de WordPress et de son écosystème

C'est donc dans ce contexte que j'ai commencé ma première semaine de stage, qui a surtout été consacrée à la prise en main de **WordPress** et la mise en place d'un environnement de travail solide. Plutôt que de me lancer directement dans le développement, j'ai pris le temps de bien comprendre le cadre technique et les outils que j'allais utiliser.

L'objectif de cette phase préparatoire était : installer, configurer et maîtriser mon environnement de développement. Pour cela, j'ai utilisé **Local**, un outil largement utilisé par la communauté **WordPress**.

J'ai mis en place un environnement basé sur **Nginx** pour le serveur, **MySQL** pour la base de données (gérée via **AdminerEvo**), et une installation propre de **WordPress**. Ce setup m'a permis de démarrer sur de bonnes bases.

Dès lors, j'ai engagé une exploration approfondie de **WordPress**. Cette phase m'a permis de comprendre la structure des fichiers, la hiérarchie des **templates** (**single.php**, **page.php**, **archive.php**, etc.), ainsi que le rôle fondamental des **hooks** (actions et filters) dans l'extensibilité du **CMS**. J'ai également étudié le fonctionnement des thèmes et les standards attendus.

### Outils de gestion de projet et mise en place des outils front-end

Au début, j'avais mis en place Git via GitHub pour le suivi du projet. Cependant, pour des raisons de confidentialité, la collectivité a préféré utiliser son propre serveur, et nous avons donc transféré le dépôt sur un serveur interne de la mairie. Pour la gestion des tâches, ma tutrice a instauré l'utilisation de ClickUp, que je consultais régulièrement pour suivre mon travail. J'ai commencé à travailler avec **TailwindCSS**, un Framework CSS utilitaire qui permet de styliser les composants de manière efficace et cohérente. Sa logique de classes utilitaires demande une certaine prise en main. J'ai ainsi testé différentes combinaisons pour styliser des éléments et intégrer des animations légères qui renforceront plus tard l'interactivité du site. Cette phase a également été l'occasion de consolider mes connaissances en **CSS**, afin de mieux contrôler les éléments non pris en charge directement par **Tailwind**, notamment la gestion fine du responsive et les comportements dynamiques. Dans la même logique, j'ai révisé les fondamentaux de **PHP** dans un contexte **WordPress** : création de fonctions personnalisées, gestion des boucles (**The Loop**), conditions, appels dynamiques de contenu avec `get_template_part()` ou `wp_query()`.

Ceci m'a permis d'entamer la structuration des fichiers clés d'un thème WordPress : `functions.php`, `header.php`, `footer.php`, ainsi que la base de templates HTML/PHP qui constituera la structure du futur site.

Un des premiers défis techniques fut l'intégration propre de TailwindCSS dans WordPress. En effet, WordPress possède une gestion de styles et de scripts très spécifique, et l'utilisation de Tailwind via PostCSS nécessite de bien configurer l'environnement pour éviter les conflits ou les chargements redondants.

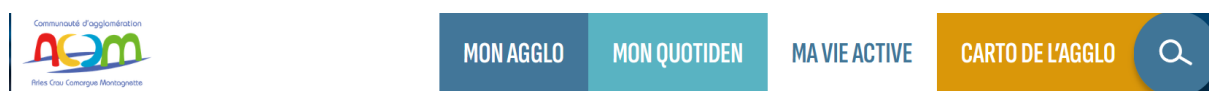
Une autre difficulté a été la mise en place des menus dynamiques, utilisant `wp_nav_menu()` et des fonctions associées pour obtenir une navigation fluide, personnalisable, et conforme à la maquette validée. Ce travail m'a amené à manipuler les structures conditionnelles et à faire un premier pas vers la personnalisation avancée des comportements natifs de WordPress.

Cette semaine a marqué une phase de montée en compétence. J'ai pu mobiliser à la fois mes acquis académiques en développement web, mais aussi en apprendre davantage sur les logiques spécifiques de WordPress et les bonnes pratiques de structuration de thème. Cela m'a permis de passer d'un état de compréhension théorique à une posture de développeur actif, capable d'organiser son environnement de travail pour accueillir les prochaines étapes du projet.

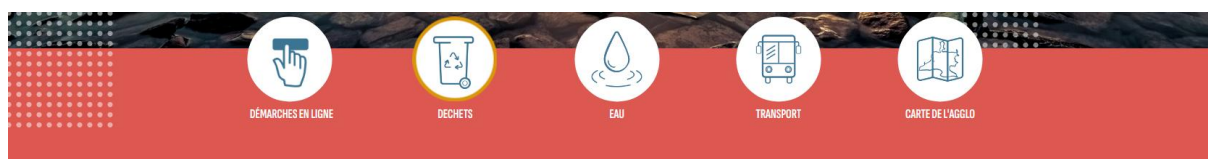
## Premiers pas vers l'intégration visuelle

Dans la continuité de cette première prise en main, la deuxième semaine de stage a été consacrée à la mise en forme de la page d'accueil. L'objectif était d'initier la construction visuelle du site, en traduisant les premiers éléments de la maquette graphique dans une structure HTML stylisée avec TailwindCSS. À ce stade, aucun contenu n'était encore relié à la base de données : l'ensemble des sections a été codé en dur, dans le seul but d'obtenir un aperçu fidèle du rendu visuel souhaité.

Le fichier `front-page.php` a été le point de départ de ce travail. J'y ai structuré les différentes zones de la page, notamment une bannière Header :



Des blocs cliquables d'accès rapide :



Et un espace réservé à l'affichage des **dernières actualités** qui aujourd'hui ressemble à ceci :



Tous ces éléments ont été intégrés manuellement, sans aucune liaison avec le **back-office WordPress**, afin de me concentrer pleinement sur la traduction des maquettes en **HTML** et la gestion du positionnement des composants.

L'un des défis majeurs rencontrés concernait l'agencement des blocs et la gestion de leur alignement, notamment dans une optique responsive. J'ai dû adapter certaines classes utilitaires de **Tailwind** pour obtenir des espacements cohérents entre les éléments, tout en veillant à conserver une hiérarchie visuelle claire. J'ai également commencé à intégrer de petites animations **CSS** pour améliorer l'interactivité de certains composants au survol, on peut le voir sur le menu « Mon Quotidien » dans le header, et sur « Déchets » dans les accès rapide.

Même si cette semaine n'a pas encore été marquée par l'introduction de données dynamiques, elle a été essentielle pour valider les choix de structure et s'assurer que l'interface visuelle était conforme aux attentes. Ce travail statique a permis de servir de maquette fonctionnelle afin de mieux visualiser l'orientation du projet, et faciliter les discussions avec ma tutrice de stage sur les améliorations à venir.

## Intégration des maquettes et découverte des outils avancés

La troisième semaine de stage a été consacrée à l'amélioration du **back-office**.

Cette semaine a commencé par un échange avec la designeuse Éléonore Dherbecourt, au cours de laquelle les maquettes graphiques 'ordinateur, tablette et mobile' ont été présentées.

Ces maquettes ont permis d'orienter le projet dans une direction visuelle.

Suite à ça, j'ai travaillé sur l'intégration des « chapeaux de page » qui sont des encadrés situés en haut de chaque page et qui permettent de contextualiser rapidement leur contenu.



Pour rendre ces éléments éditables, j'ai découvert et pris en main le **plugin Advanced Custom Fields (ACF)**, recommandé par ma tutrice et par Frédéric Cases, développeur du service. Ce **plugin** permet de créer des champs personnalisés dans l'administration **WordPress** et d'y accéder en front via du code **PHP** dans les **templates**.

J'ai donc défini des champs textuels :

Articles  
Médias  
Pages  
Réglages Accueil  
Apparence  
Extensions 2  
Comptes  
Outils  
Réglages  
ACF  
Groupes de champs  
Types de publication  
Taxonomies  
Pages d'options  
Outils  
FileBird  
Réduire le menu

1	▼	chapeau de page	chapeau_de_page	Case à cocher
2	▼	url du pictogramme	pictogramme	Liste déroulante
3	▼	sous titre	sous_titre	Texte
4	▼	paragraphe	paragraphe	Éditeur WYSIWYG
5	▼	Bloc Info	bloc_info	Case à cocher
6	▼	service *	service	Texte
7	▼	adresse *	adresse	Éditeur WYSIWYG
8	▼	horaires *	horaires	Éditeur WYSIWYG
9	▼	Bas du Bloc info	bas_du_bloc_info	Bouton radio
10	▼	numéro *	numero	Texte
11	▼	lien *	lien	Lien

+ Ajouter un champ

Ces champs peuvent être remplis par un utilisateur avec le rôle « éditeur » afin de rendre l'interface plus autonome et intuitive, ils apparaissent désormais sur l'édition des pages :

Les bons gestes de tri - Page
Ctrl+K

chapeau de page
^ v

Voulez vous un chapeau de page ?

☒ OUI
☐ NON

sous titre (si le paragraphe est disponible)

LE TRI, UN PETIT GESTE AU GRAND EFFETS !

Picto

Celui de la catégorie

Aucune image sélectionnée

paragraphe

Ajouter un média

Visuel
Texte

Paragraphe
B I
Liste à puces
Liste numérotée
Citation
Alignement à gauche
Alignement au centre
Alignement à droite
Liens
Tableaux

Le tri n'est pas qu'un petit geste, son impact est bien plus vaste qu'il n'y paraît.

- **Trier**, c'est préserver nos ressources naturelles
- **Trier**, c'est limiter les émissions de gaz à effet de serre
- **Trier**, c'est économiser de l'énergie
- **Trier**, c'est soutenir l'économie et l'emploi localement

Aujourd'hui, 72% des emballages ménagers et 62 % des papiers sont recyclés grâce au geste de tri des Français, devenu premier geste éco-citoyen. Plusieurs spots publicitaires signés Citéo, nous encouragent toutes et tous à poursuivre cet effort collectif.

L’affichage dynamique de ces champs a été mis en œuvre dans le thème WordPress, via une adaptation du fichier `page.php` grâce à l’utilisation de la méthode fournie par ACF « `get_field()` », permettant de récupérer la valeur des champs d’une page grâce à leurs noms techniques définis lors de leurs créations.

```

<main id="primary" class="site-main container page-contenu">
  <section class="chapeau">
    <h1><?php the_title(); ?></h1>
    <div class="chapeau-infos">
      <?php
      if (get_field('chapeau_de_page', $page_id) && strtoupper(get_field('chapeau_de_page'

// Affiche le pictogramme
$liste_pictos = get_field('liste_pictos', $page_id);
if (!empty($liste_pictos)) {
    echo '<div class="picto"><img src="" . esc_url($liste_pictos) . "" alt="Pict
} elseif (!empty($term_picto)) {
    echo '<div class="picto"><img src="" . esc_url($pathImgIcons . '/' . $term_p
}

```

Cette étape a nécessité l'apprentissage et l'implémentation de **hooks WordPress** intégré au fichier **functions.php**, permettant de mieux contrôler l'injection de contenu dans le cycle de vie du thème. J'ai par exemple utilisé le **hook** : « **wp\_enqueue\_scripts** » pour lier mon fichier **CSS** de sortie.

Parallèlement, la validation de l'arborescence du site a permis de me projeter dans l'architecture fonctionnelle à venir.

J'ai pu ainsi me documenter sur la création de blocs Gutenberg personnalisés, permettant l'utilisation de composants flexibles et réutilisables.

Enfin, l'utilisation des **template-parts** a permis d'avoir un code mieux organisé, où chaque partie de la page est isolée dans un fichier spécifique, facilitant les futures modifications et la maintenance du projet.

## Structuration des contenus et premières logiques dynamiques

Cette quatrième semaine a été marquée par la structuration des contenus. Il ne s'agissait plus uniquement d'intégrer des éléments visuels ou de manipuler des **templates** génériques, mais de réfléchir à l'architecture du site **WordPress** en fonction des besoins réels exprimés par la collectivité et des contraintes posées par la maquette graphique.

Après plusieurs échanges avec ma tutrice et Frédéric Cases, il est décidé de conserver l'usage natif des **articles WordPress** pour les actualités, tout en structurant les sous-sections à l'aide de catégories. Ce choix technique permet de garder un fonctionnement simple et adapté aux utilisateurs.

D'autre part, j'ai entamé la construction d'un système de contenus personnalisés pour des sections plus spécifiques du site comme les pages. Cela a pris la forme d'un **CPT** (**Custom Post Type**) intitulé « Mon Quotidien ». Ce **CPT** a été complété par la création d'une **taxonomie** personnalisée « Mes Déchets », elle-même hiérarchisée en sous-termes (ex : Je Trie, Les Bons Gestes de Tri, etc.). Deux autres **CPT** sont créés pour les menus « Mon Agglo » et « Ma vie active » ainsi que leurs **taxonomies**.

Sur le plan technique, j'ai également amorcé la création du **template single.php** dédié aux **CPT**. En complément de tout ça, j'ai mis en place une **page de menu** dans l'administration pour permettre la modification dynamique de certains éléments clés du site, notamment **l'image de fond de la page d'accueil**. Cela a nécessité l'utilisation de fonctions telles que :

`add_menu_page()`

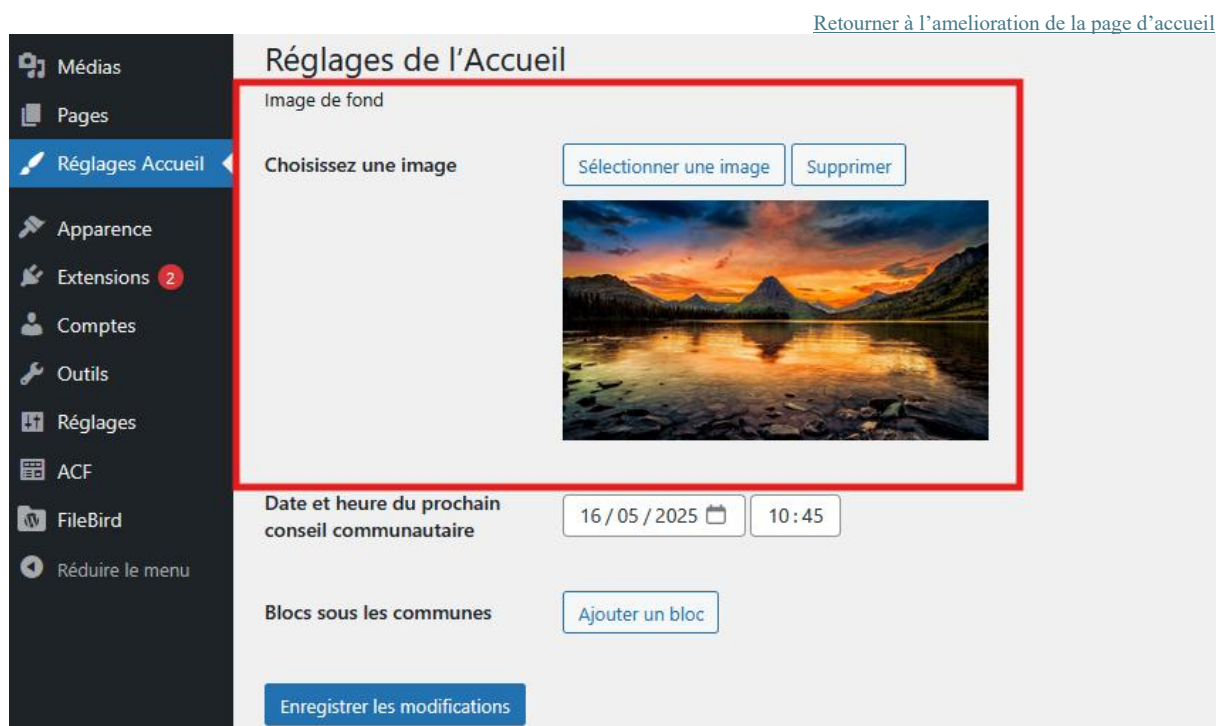
```
1070     function add_home_settings_page()
1071     {
1072         add_menu_page(
1073             'Réglages Accueil',          // Nom de la page
1074             'Réglages Accueil',          // Nom dans le menu
1075             'edit_posts',                 // Accessible aux éditeurs
1076             'home-settings',              // Slug de la page
1077             'home_settings_page_html',    // Fonction d'affichage
1078             'dashicons-admin-customizer', // Icône du menu
1079             30                             // Position dans le menu
1080         );
1081     }
1082     add_action('admin_menu', 'add_home_settings_page');
```

`add_settings_field()`

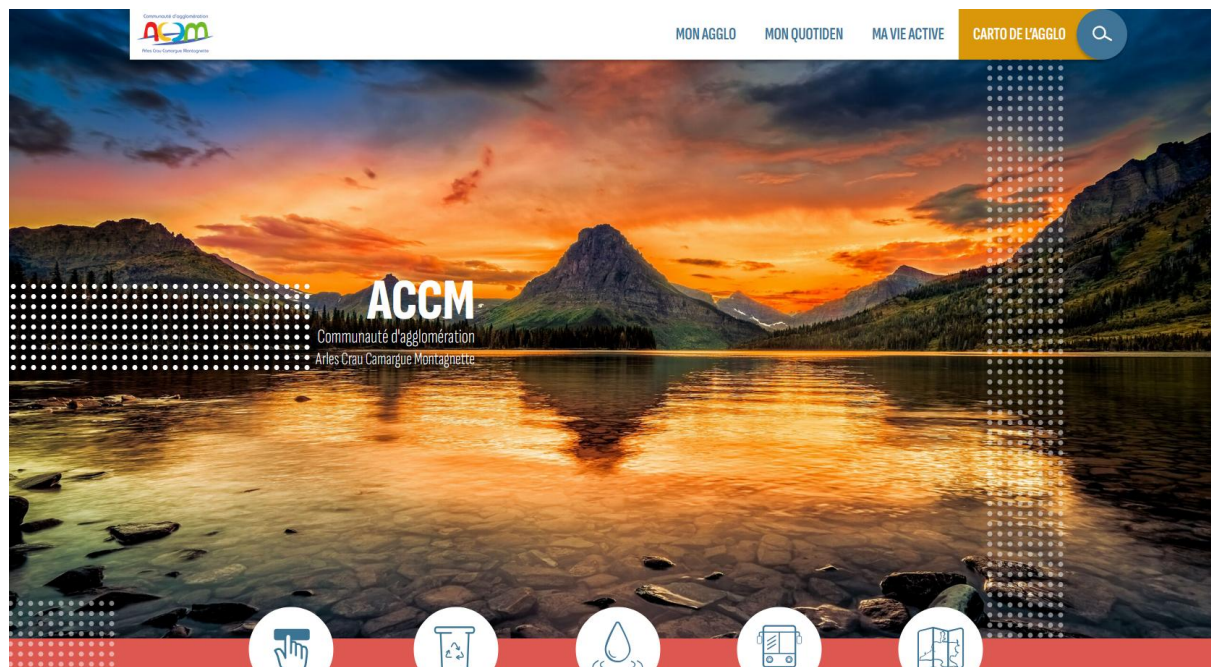
```
// Ajoute un champ de paramètre à la page de réglages personnalisée
add_settings_field(
    'home_background_image',             // ID du champ (utilisé pour l'identification)
    'Choisissez une image',               // Titre affiché à côté du champ
    'home_background_image_callback',     // Fonction de rappel qui affiche le champ
    'home-settings',                     // Slug de la page où ce champ sera affiché
    'home_settings_section'              // ID de la section dans laquelle le champ sera placé
);
```

Ou encore `wp.media()` pour utiliser la **médiathèque WordPress**.

J'ai ainsi créé une interface simple et accessible, et intuitive pour les utilisateurs.



Pour un résultat en front-end comme ceci :



Résultat avec une image fictive

D'un point de vue humain, cette semaine a également renforcé ma collaboration avec l'équipe par les retours du designer sur les animations **CSS**, permettant d'ajuster mes styles de manière plus précise.

## Restructuration des contenus et personnalisation de l'éditeur Gutenberg

La semaine précédente a permis de mettre en place une structure de contenus reposant sur un système de **Custom Post Types** et de **taxonomies**. Cependant, après un réexamen collectif du fonctionnement et des besoins réels du service communication, une réorientation structurelle a été décidée. Le principe des trois **CPT** n'a finalement pas été retenu. La décision a été prise de revenir à une gestion plus classique des contenus, par l'utilisation des pages **WordPress**, enrichies d'une **taxonomie** personnalisée baptisée « custom-categories ». Celle-ci permettra de gérer l'arborescence des pages (catégories, sous-catégories, etc.).

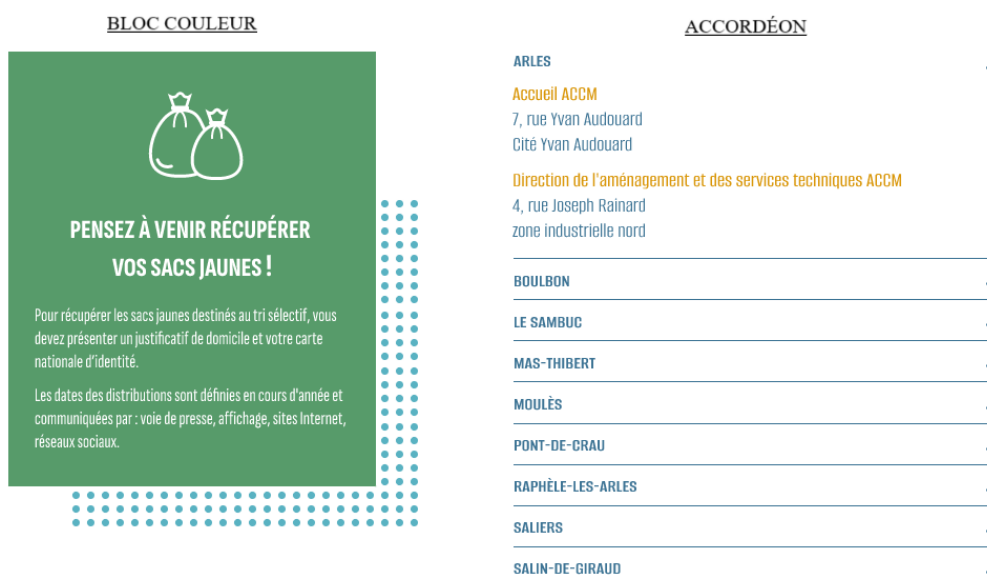
```
28 function accm_setup()  
228 // Taxonomies ACCM  
229 function create_page_taxonomy()  
230 {  
231     $args = array(  
232         'hierarchical' => true, // Définit la taxonomie comme hiérarchique  
233         'labels' => array( // Définition des libellés visibles dans l'administration  
234             'name' => 'categories', // Nom au pluriel  
235             'singular_name' => 'categorie', // Nom au singulier  
236             'search_items' => 'Rechercher les categories', // Texte du champ de recherche  
237             'all_items' => 'Toutes les categories', // Libellé pour la liste complète  
238             'parent_item' => 'parent', // Libellé pour l'élément parent  
239             'parent_item_colon' => 'parent : ', // Variante avec deux-points pour l'interface  
240             'edit_item' => 'Modifier la categorie', // Libellé pour l'action de modification  
241             'update_item' => 'Mettre à jour la categorie', // Libellé pour l'action de mise à jour  
242             'add_new_item' => 'Ajouter une nouvelle categorie', // Libellé pour l'ajout d'un nouvel élément  
243             'new_item_name' => 'Nom de la nouvelle categorie', // Libellé pour le nom d'un nouvel élément  
244             'menu_name' => 'Categories', // Nom affiché dans le menu d'administration  
245         ),  
246         'rewrite' => array( // Définition des règles de réécriture d'URL  
247             'slug' => '', // Aucun slug (pourquoi? -> dans la suite)  
248             'with_front' => false, // Désactive le préfixe de base  
249             'hierarchical' => true, // Les URLs peuvent refléter la hiérarchie  
250         ),  
251         'show_ui' => true, // Affiche l'interface d'administration pour cette taxonomie  
252         'show_in_menu' => true, // Affiche cette taxonomie dans les menus de l'admin  
253         'show_in_rest' => true, // Active cette taxonomie pour Gutenberg  
254     );  
255  
256     // Enregistre la taxonomie "custom-categories" pour le type de contenu "page"  
257     register_taxonomy('custom-categories', 'page', $args);
```

Ce changement d'approche a nécessité une reconfiguration complète de la structure du site. Du point de vue technique, cette solution permet de tirer parti des fonctionnalités internes du **CMS** en conservant une structure logique et intuitive pour les agents en charge de l'édition.

On a donc maintenant une **taxonomie** « Categories » pour les pages comme pour les articles :



Parallèlement à cette phase de réajustement, J'ai commencé le développement des **blocs Gutenberg**. Deux blocs étaient au cœur des priorités de cette semaine : le bloc « Couleur » et le bloc « Accordéon ».



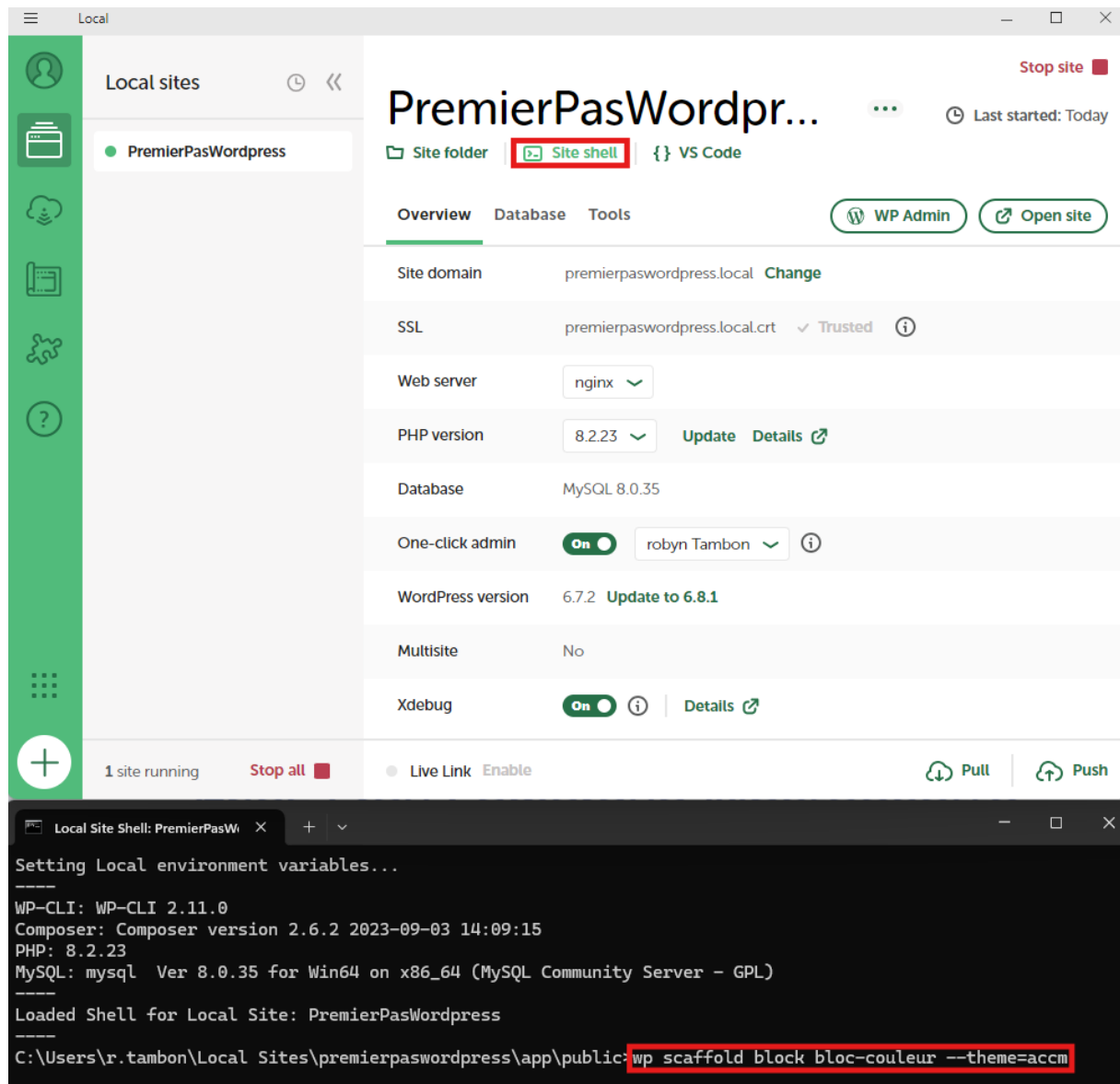
Le bloc « Couleur » se compose de trois éléments principaux : un pictogramme, un sous-titre et un paragraphe. Développé en **JSX** avec **React**, ce bloc est directement utilisable dans l'éditeur **Gutenberg** et s'affiche dynamiquement en front-end.

## Étapes de la création de bloc custom

### *Génération du squelette du bloc via WP-CLI*

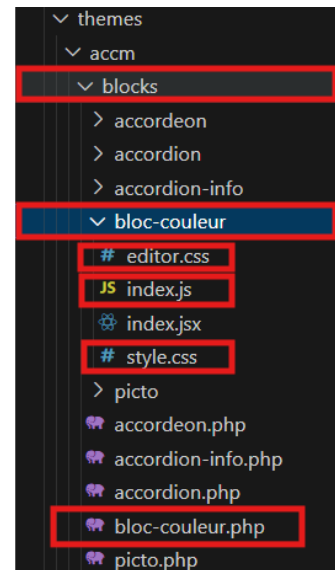
J'ai utilisé l'interface de commande de WordPress, **WP-CLI**, qui permet d'automatiser la génération des fichiers de base. Dans le terminal, disponible dans le logiciel **Local**, la commande suivante a été exécutée :

```
wp scaffold block bloc-couleur --theme=accm
```



Cette commande a généré un dossier `blocks/bloc-couleur/` contenant l'architecture du bloc :

- Fichier `PHP` d'enregistrement
- Fichier JavaScript (`index.js`) pour l'interface de l'éditeur,
- Et fichiers de `style` (optionnels).



C'est cette structure que j'ai ensuite enrichie.

### *Mise en place de Babel*

Le bloc étant développé en `JSX` avec `React`, il a fallu traduire ce code vers du `JavaScript` compatible navigateurs. Pour cela, j'ai installé `Babel` et configuré le fichier « `.babelrc` » à la racine de mon thème :

```
wp-content > themes > accm > .babelrc > ...
1  {
2    "presets": [
3      ["@babel/preset-env"]
4    ],
5    "plugins": [
6      ["@babel/plugin-transform-react-jsx", {"pragma": "wp.element.createElement"}],
7      ["@babel/plugin-syntax-object-rest-spread"],
8      ["@babel/plugin-transform-runtime"]
9    ]
10 }
11
```

`@babel/preset-env` : traduit le `JS` moderne (`JSX`) en un `JS` lisible par tous les navigateurs.

`@babel/plugin-transform-react-jsx` : transforme le `JSX` en syntaxe compatible avec `WordPress` (`wp.element.createElement`).

Les autres `plugins` permettent de gérer des syntaxes avancées (objets étendus, `async/await`...).

## Configuration de Webpack

Pour compiler le code **JSX** en **JS** final utilisable par **WordPress**, j'ai configuré **Webpack** avec ce fichier **webpack.config.js** :

```
wp-content > themes > accm > webpack.config.js > ...
1  const path = require('path');
2
3  module.exports = {
4    entry: './blocks/bloc-couleur/index.jsx',
5    output: {
6      path: path.resolve(__dirname, 'blocks/bloc-couleur'),
7      filename: 'index.js'
8    },
9    module: {
10     rules: [
11       {
12         test: /\.jsx?$/,
13         loader: 'babel-loader',
14         exclude: /node_modules/,
15       }
16     ]
17   },
18   mode: 'development',
19 }
20
```

Grâce à cette configuration, j'ai pu exécuter la commande suivante dans le terminal pour générer un fichier **index.js** exploitable dans **WordPress** :

npx webpack

## Dynamisme des blocs : pourquoi ce choix ?

J'ai opté pour un bloc dynamique plutôt que statique. Cela signifie que le rendu du bloc est généré côté serveur (**PHP**) via une fonction **render\_callback**, et non enregistré directement dans le contenu **HTML**. Ce choix présente plusieurs avantages dans le contexte d'une collectivité :

si on modifie le rendu **HTML**, tous les blocs existants sont mis à jour automatiquement.

Intégration dynamique de données.

Meilleur contrôle du rendu pour respecter la charte graphique institutionnelle.

## Structure du fichier JSX

Le fichier `index.jsx` contient toute la logique du bloc, depuis son enregistrement jusqu'à l'interface utilisateur dans Gutenberg. Voici un aperçu des composants principaux :

wp-content > themes > accm > blocks > bloc-couleur > index.jsx > ...

```
1  const { __ } = wp.i18n;
2  const { registerBlockType } = wp.blocks;
3  const { InspectorControls, RichText, BlockControls, AlignmentToolbar } = wp.editor;
4  const { PanelBody, TextControl, SelectControl, RadioControl } = wp.components;
5
```

Ces modules permettent :

1. `registerBlockType()` : d'enregistrer un bloc sous le nom accm/bloc-couleur.

```
6  registerBlockType('accm/bloc-couleur', {
7    title: __('Accm - Bloc Couleur', 'accm'),
8    icon: 'admin-site',
9    category: 'widgets',
10   attributes: {
11     title: { type: 'string', default: '' },
12     text: { type: 'string', default: '' },
13     picto: { type: 'string', default: data && data.defaultPicto ? data.defaultPicto : '' },
14     color: { type: 'string', default: 'vert' },
15     align: { type: 'string', default: 'center' },
16   },
17 }
```

2. `InspectorControls` : de gérer les paramètres dans la colonne latérale (titre, pictogramme, couleur...).
3. `BlockControls` : d'ajouter des options au-dessus du bloc (par ex : alignement).
4. `RichText`, `TextControl` : champs pour la saisie de texte enrichi ou simple.
5. `SelectControl` : pour sélectionner un pictogramme.
6. `RadioControl` : pour choisir la couleur du bloc (vert ou rouge).

Les données sont enregistrées dans les attributs du bloc (titre, texte, pictogramme, couleur...), puis transmises à la fonction PHP de rendu via la méthode `save: () => null`, ce qui signifie que le bloc est 100 % dynamique.

```
90  // On utilise un render_callback PHP, donc rien n'est sauvegardé côté HTML.
91  save: () => {
92    return null;
93  },
```

## Enregistrement et rendu côté PHP

Enfin, dans le fichier `bloc-couleur.php`, j'ai défini les fonctions permettant à WordPress de :

- Charger le script généré (`index.js`) dans l'éditeur.
- Enregistrer le bloc via `register_block_type()`.
- Afficher dynamiquement le HTML dans le front grâce à `render_callback`.

La structure ressemble à ceci :

```
16 function bloc_couleur_block_init()  
17 {  
18     // Skip block registration if Gutenberg is not enabled/merged.  
19     if (! function_exists('register_block_type')) return;  
20     $dir = get_stylesheet_directory() . '/blocks';  
21     $index_js = 'bloc-couleur/index.js';  
22     wp_register_script(  
23         'bloc-couleur-block-editor',  
24         get_stylesheet_directory_uri() . "/blocks/{$index_js}",  
25         [  
26             'wp-blocks',  
27             'wp-i18n',  
28             'wp-element',  
29             'wp-components',  
30             'wp-editor'  
31         ],  
32         filemtime("{$dir}/{$index_js}")  
33     );  
34     $path_img = site_url() . '/wp-content/themes/accm/img';  
35     register_block_type('accm/bloc-couleur', [  
36         'editor_script' => 'bloc-couleur-block-editor',  
37         'editor_style' => 'bloc-couleur-block-editor',  
38         'style' => 'bloc-couleur-block',  
39         'attributes' => [  
40             'title' => [  
41                 'type' => 'string',  
42                 'default' => ''  
43             ],  
44             'text' => [  
45                 'type' => 'string',  
46                 'default' => ''  
47             ],  
48             'picto' => [  
49                 'type' => 'string',  
50                 'default' => reset(accm_get_pictos())  
51             ],  
52             'color' => [  
53                 'type' => 'string',  
54                 'default' => 'vert'  
55             ]  
56         ],  
57         'render_callback' => 'bloc_couleur_block_render',  
58     ]);  
59 }  
60 add_action('init', 'bloc_couleur_block_init');
```

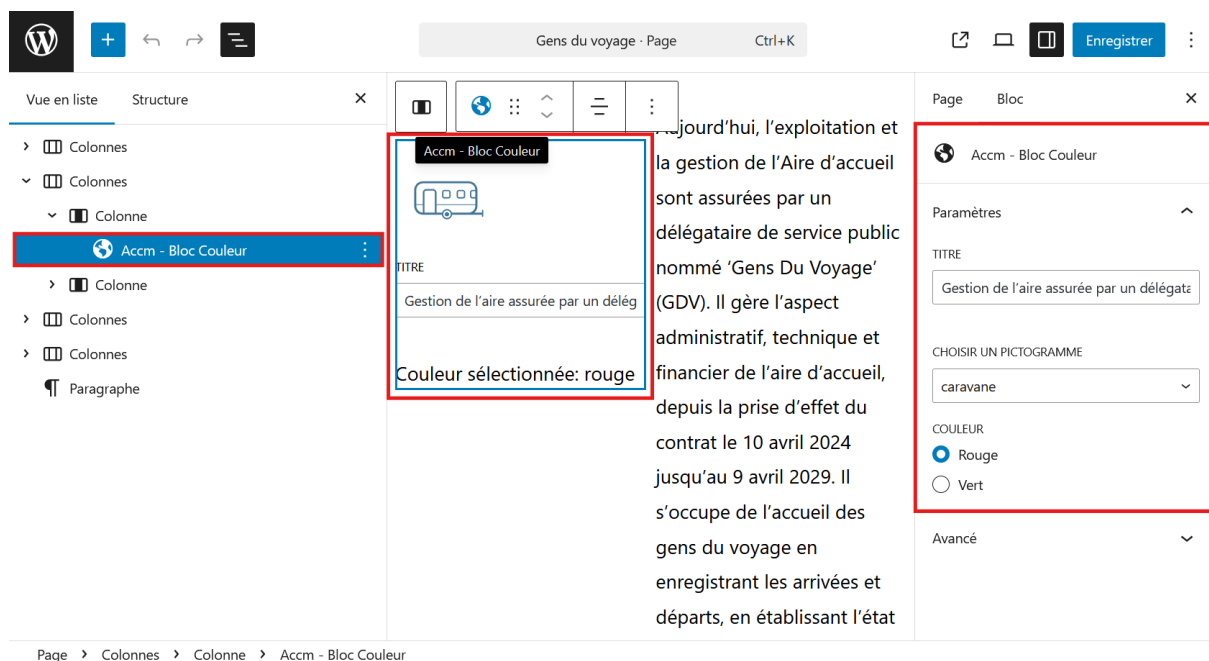
Le `render_callback bloc_couleur_block_render()` construit le **HTML** côté serveur, en utilisant les attributs du bloc, et insère dynamiquement les pictogrammes **SVG**, en tenant compte de la couleur choisie (rouge ou vert). Cette méthode permet aussi de modifier dynamiquement le contenu **SVG**.

```

63  /**
64   * Génère le HTML du bloc côté front (render_callback).
65   *
66   * @param array $attributes Attributs du bloc
67   * @return string
68   */
69  function bloc_couleur_block_render($attributes)
70  {
71      $title = isset($attributes['title']) ? esc_html($attributes['title']) : '';
72      $text = isset($attributes['text']) ? wp_kses_post($attributes['text']) : '';
73      $picto = isset($attributes['picto']) ? esc_url($attributes['picto']) : '';
74      $color = isset($attributes['color']) ? esc_attr($attributes['color']) : 'vert';
75      $align = isset($attributes['align']) ? esc_attr($attributes['align']) : 'center';
76      $alignStyle = '';
77      if ($align === 'left') {
78          $alignStyle = "justify-content: flex-start;";
79      } elseif ($align === 'right') {
80          $alignStyle = "justify-content: flex-end;";
81      } else {
82          $alignStyle = "justify-content: center;";
83      }
84      $svgContent = file_get_contents($picto);
85      // Remplacer toutes les classes "st0", "st1" et "st2" par un fill="white"
86      if (str_ends_with($picto, 'emission-gaz.svg')) {
87          $svgContent = preg_replace('/class="st0"/i', 'fill="white"', $svgContent);
88          $svgContent = preg_replace('/class="st1"/i', 'fill="white"', $svgContent);
89      } elseif (str_ends_with($picto, 'budget.svg')) {
90          $svgContent = preg_replace('/class="st0"/i', 'stroke="white" stroke-width="3.9"', $svgContent);
91          $svgContent = preg_replace('/class="st1"/i', 'stroke="white" stroke-width="2.9"', $svgContent);
92          $svgContent = preg_replace('/class="st2"/i', 'fill="white"', $svgContent);
93      } elseif (str_ends_with($picto, 'ici.svg')) {
94          if ($color == 'vert') {
95              $svgContent = preg_replace('/class="st0"/i', 'fill="#579B6A"', $svgContent);
96          } elseif ($color == 'rouge') {
97              $svgContent = preg_replace('/class="st0"/i', 'fill="#DD5851"', $svgContent);
98          }
99          $svgContent = preg_replace('/class="st1"/i', 'fill="white"', $svgContent);
100      } else {
101          $svgContent = preg_replace('/class="st0"/i', 'fill="white"', $svgContent);
102      }
103      ob_start();
104      <div style="display: flex; margin-bottom: 2.6rem; <?= $alignStyle ?>">
105          <article class="bloc-couleur <?= $color ?>">
106              <?php if ($picto) : ?>
107                  <div class="picto">
108                      <?= $svgContent ?>
109                  </div>
110              <?php endif; ?>
111              <h2><?= $title ?></h2>
112              <p><?= $text ?></p>
113          </article>
114      </div><?php
115      return ob_get_clean();
116  }

```

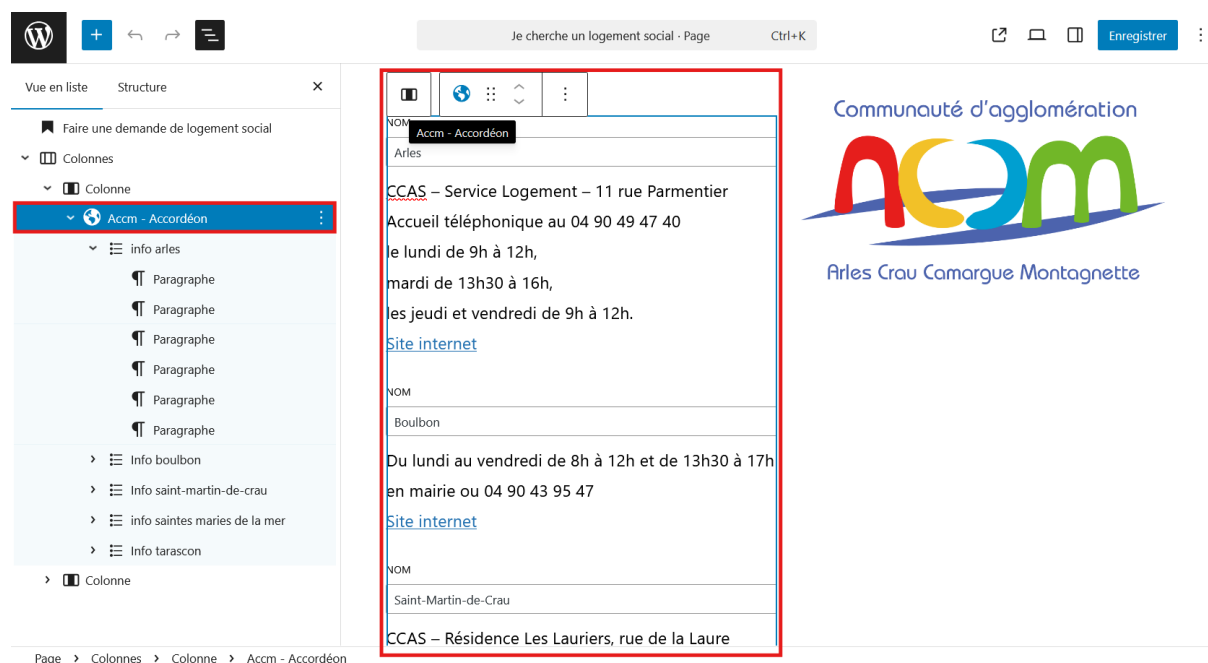
Avec tout ce travail, on obtient maintenant ce nouveau bloc dans l'éditeur Gutenberg :



Pour un affichage conforme à l'identité graphique :



J'ai aussi effectué un travail important de refonte du bloc « Accordéon » parce que les modifications d'un accordéon affectaient toutes les communes de ce dernier, ce qui n'était pas concevable. Pour corriger cela, j'ai mis en place un bloc parent « accordion » capable de contenir plusieurs blocs enfants « accordion-info », chacun encapsulant son propre contenu via un système d'InnerBlocks.



Cette structure permet désormais d'associer de manière indépendante des informations spécifiques à chaque commune.

Pour un résultat tel que :



D'autre part, un bloc dédié « Communes » a été développé pour intégrer des informations détaillées telles que la superficie, le nom ou la population, il devait effectuer un rendu tel que :

## ARLES



51 840 habitants



758,93 Km²



### MAIRIE

Place de la République

13200 Arles

06 90 49 36 36

SITE WEB



---

## BOULBON



---

## SAINT-MARTIN DE CRAU



---

## SAINT-PIERRE-DE-MÉZOARGUES



---

## TARASCON



Mais il a été abandonné par la suite pour cause qu'on devait pouvoir effectuer ce rendu avec le bloc **accordion**, en effet, avec l'évolution apportée à ce dernier grâce au composant **InnerBlocks**, il peut maintenant contenir plusieurs bloc enfants de tous les types, on peut donc les agencer comme on le souhaite.

## Gestion des pictogrammes et autres champs des custom-categories

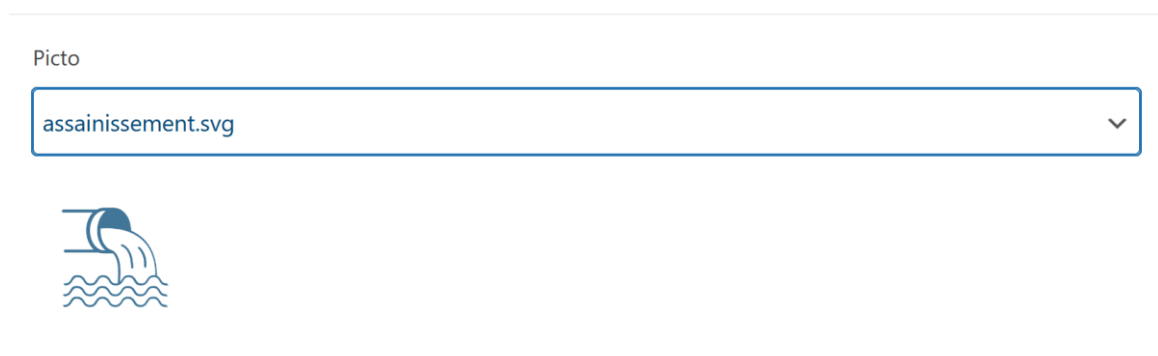
Une gestion des pictogrammes a aussi été mise en place. J'ai créé une fonction **PHP** pour récupérer automatiquement les fichiers **SVG**, **JPG**, **PNG** du répertoire dédié



Et une autre fonction pour les afficher dans une liste déroulante **ACF** dans l'éditeur **Gutenberg**.

```
558 function accm_remplir_acf_select_pictos($field)
559 {
560     $pictos = accm_get_pictos();
561     if ($pictos) {
562         $field['choices'] = array();
563         $field['choices'][''] = 'Celui de la catégorie';
564         foreach ($pictos as $filename => $url) {
565             $field['choices'][$url] = $filename;
566         }
567     }
568     return $field;
569 }
570 add_filter('acf/load_field/name=liste_pictos', 'accm_remplir_acf_select_pictos');
```

On obtient cet affichage :



La logique d'héritage hiérarchique a également été implémentée : si un terme enfant ne dispose pas de pictogramme, celui-ci est hérité dynamiquement du parent via une boucle `while` permettant de remonter la hiérarchie jusqu'à trouver une icône disponible.

```

906         if (!empty($parent_term) && !is_wp_error($parent_term)) {
907             while (!empty($parent_term) && !is_wp_error($parent_term)) {
908                 $parent_picto = get_term_meta($parent_term->term_id, 'term_picto', true);
909                 if (!empty($parent_picto) && empty($term_picto)) {
910                     $term_picto = $parent_picto;
911                     $isParentPicto = true;
912                     break;
913                 }
914                 $parent_term = get_term($parent_term->parent, $parent_term->taxonomy);
915             }
916         }

```

Cette fonctionnalité a été cruciale pour assurer une cohérence graphique sur l'ensemble du site sans imposer une saisie redondante à chaque terme de `taxonomie`.

Enfin, la `taxonomie` a été renforcée par l'ajout de contrôles de saisie lors de la validation. J'ai intégré une vérification stricte : chaque terme de catégorie doit être associé à un pictogramme. Si aucun pictogramme n'est sélectionné et que le terme n'a aucun parent, une erreur `WP_Error` est renvoyée pour éviter les oublis.

```

1011 function accm_validate_term_picto($term, $taxonomy)
1012 {
1013     $allowed_taxonomies = ['custom-categories'];
1014     if (in_array($taxonomy, $allowed_taxonomies)) {
1015         $parent = isset($_POST['parent']) ? sanitize_text_field($_POST['parent']) : '';
1016         $term_picto = isset($_POST['term_picto']) ? sanitize_text_field($_POST['term_picto']) : '';
1017         $term_slug = isset($_POST['slug']) ? sanitize_text_field($_POST['slug']) : '';
1018
1019         // Si le terme n'a "Aucun" parent et qu'aucun pictogramme n'est sélectionné
1020         if ($parent === '-1' && empty($term_picto)) {
1021             return new WP_Error('term_picto_error',
1022                 "Un pictogramme doit être sélectionné pour une catégorie qui n'a pas de parent.");
1023         }
1024     }
1025     return $term;
1026 }
1027 add_filter('pre_insert_term', 'accm_validate_term_picto', 10, 2);

```

Des champs enrichis (nom, **Slug**, Picto, Lien Externe) ont été ajoutés à chaque terme via des **hooks** personnalisés :

1. `custom-categories_add_form_fields` pour l'ajout de catégorie de page
2. `custom-categories_edit_form_fields` pour l'édition de terme de **taxonomie**

améliorant à la fois le contrôle qualité et la richesse des contenus.

```

837 // Ajout du champ lors de la création d'un terme
838 add_action('custom-categories_add_form_fields', function () {
839     ?>
840     <div class="form-field">
841         <label for="term-url">Lien Externe</label>
842         <input type="url" name="term_url" id="term-url" value="" size="40">
843         <p class="description">Ajoutez un lien externe pour ce terme.</p>
844     </div>
845     <?php
846 });
847
848 // Ajout du champ lors de l'édition d'un terme
849 add_action('custom-categories_edit_form_fields', function ($term) {
850     $term_url = get_term_meta($term->term_id, 'term_url', true);
851     ?>
852     <tr class="form-field">
853         <th scope="row"><label for="term-url">Lien Externe</label></th>
854         <td>
855             <input type="url" name="term_url" id="term-url" value="<?php echo esc_attr($term_url); ?>" size="40">
856             <p class="description">Ajoutez un lien externe pour ce terme.</p>
857         </td>
858     </tr>
859     <?php
860 });

```

## categories

Un pictogramme doit être sélectionné pour une catégorie qui n'a pas de parent.

Rechercher les categories

Ajouter une nouvelle categorie

Nom

Ma categorie test

Le nom est la façon dont il apparaît sur votre site.

Slug

mon-memoire-de-stage

Le « slug » est la version du nom normalisée pour les URL. Il ne contient généralement que des lettres, des chiffres et des traits d'union.

parent

Aucun

Assignez un terme parent pour créer une hiérarchie. Le terme Jazz, par exemple, serait le parent de Bebop et Big Band.

Lien Externe

Ajoutez un lien externe pour ce terme.

Picto

Aucun

Voulez vous un chapeau de page ?

☒ OUI
☐ NON

Actions groupées

Appliquer

33 éléments

«

<

1

>

»

<input type="checkbox"/>	Nom	Description	Slug	Total
<input type="checkbox"/>	actualites	—	actualites	1
<input type="checkbox"/>	Commerçant artisan	—	commercant-artisan	4
<input type="checkbox"/>	Demandeur d'emploi	—	demandeur-demploi	2
<input type="checkbox"/>	Entrepreneur	—	entrepreneur	6
<input type="checkbox"/>	Etudiant	—	etudiant	2
<input type="checkbox"/>	L'eau	—	leau	0
<input type="checkbox"/>	— Mon assainissement	—	mon-assainissement	2

Résultat dans le back-Office WP

TAMBON Robyn

Mémoire de stage 2024-2025

Page 32 sur 63

D'un point de vue technique, cette semaine a été particulièrement dense et formatrice. J'ai dû manipuler des outils complexes (**Webpack**, **Babel**, **React**), tout en assurant la compatibilité de mes développements avec **WordPress**. Cette phase de travail m'a permis de mieux comprendre le fonctionnement internes de **Gutenberg**, et de renforcer ma capacité à produire des composants réutilisables.

Sur le plan humain, cette période a encore renforcé les échanges avec ma tutrice, très impliquée dans la validation de la structure, et avec Frederic Cases, dont les conseils ont été précieux sur le plan technique comme architectural. Les échanges en réunion ont permis d'ajuster progressivement le cap tout en maintenant une cohérence d'ensemble avec les attentes de l'ACCM.

## Approfondissement des templates et stabilisation des blocs personnalisés

### Amélioration du template des taxonomies

Un autre axe de cette semaine a concerné le fichier **taxonomy-custom-categories.php**, qui gère l'affichage des catégories personnalisées du site. Conformément à la maquette, j'ai développé un sommaire dynamique qui affiche de manière hiérarchique les premières sous-catégories (termes enfants de premier niveau) et les pages directement associées à la catégorie consultée. Cette fonctionnalité, cruciale pour la navigation utilisateur, a soulevé des défis techniques :

Il a fallu concevoir une logique conditionnelle robuste capable de distinguer les pages orphelines de sous-catégories, afin d'éviter les erreurs d'affichage.

Le comportement graphique du sommaire a également été peaufiné, pour correspondre au niveau d'exigence visuel attendu.



En complément, le groupe de champs **ACF** nommé "Chapeau de page" a été intégré aux termes de la **taxonomie**. J'ai également ajouté un champ **ACF** dédié au pictogramme des pages.

J'ai identifié une erreur au niveau de la case à cocher « NON » censée désactiver l'affichage du chapeau : elle n'avait aucun effet. Ce dysfonctionnement provenait d'une condition manquante dans le template, que j'ai corrigé en ajoutant une vérification explicite de l'état de la case. Ce correctif a permis de rendre les champs conditionnels réellement dynamiques.

#### Remarque :

Le **fil d'Ariane**, utilisé notamment sur les pages d'actualités et de **taxonomies**, a été modularisé dans un **template-part** par ma tutrice, afin de permettre son réemploi simplifié sur d'autres **templates**.

#### Finalisation des pages principales et navigation

Cette semaine a aussi été marquée par des ajustements sur plusieurs fichiers de base du thème:

Page d'accueil (**front-page.php**) : stylisée et complétée pour respecter les zones de contenus prévues par la maquette.

Footer (**footer.php**) : stabilisé graphiquement et fonctionnellement.

Affichage des pictogrammes : correction d'un bug où le pictogramme défini pour une page apparaissait aussi sur certaines catégories. J'ai isolé les conditions pour séparer les logiques d'affichage et rétablir la cohérence.

Ajout d'un bouton « Back to top » : intégré dans le **footer** de façon discrète mais efficace, il permet de revenir en haut de page sans rechargement.



*BTT*

*BTT Hover*

#### Création de la page Actualités

Une page dédiée aux actualités ([page-actualites.php](#)) a été créée par ma tutrice pour rassembler l'ensemble des articles du site. Conformément à la logique **UX**, lorsqu'un utilisateur clique sur le bouton **+** dans la section des actualités sur la page d'accueil, il est redirigé automatiquement vers cette page unique, afin de centraliser l'information.



*Section actualités sur la page d'accueil*

Le **fil d'Ariane** a été intégré dans ce template ainsi que dans [category.php](#) (template pour les catégories d'articles), renforçant la cohérence de navigation entre les articles.

**VOUS ÊTES ICI : ACCUEIL > MON QUOTIDIEN > MES DÉCHETS > JE TRIE > LES BONS GESTES DE TRI**

*Exemple fil d'ariane*

## Consolidation éditoriale et montée en flexibilité des blocs

La semaine a été principalement consacrée à la consolidation de la structure du site, avec une attention particulière portée au remplissage des contenus structuraux, à la finalisation de certains **blocs Gutenberg**, et à la stabilisation des interfaces visuelles dans le **back-office**.

### Structuration progressive des contenus

En s'appuyant sur la structure validée précédemment (pages + **taxonomie** personnalisée), j'ai procédé à l'ajout manuel du reste des catégories principales dans **WordPress**, une première partie ayant été déjà saisie par ma tutrice. Nous synchronisons régulièrement nos

environnements **WordPress** en partageant les contenus, ce qui nous permettait de gagner du temps tout en assurant une cohérence dans l'arborescence globale.

J'ai également créé les pages enfants associées à ces catégories. À ce stade, leur contenu est resté partiellement vide, à l'exception de quelques pages pilotes qu'on a remplies à titre d'exemple. L'objectif était de tester le rendu visuel et la logique d'affichage dans différentes configurations de contenu.

Il est important de noter que le remplissage complet de ces pages ne faisait pas partie de nos missions : cette tâche revient au service communication de l'ACCM.

En parallèle, j'ai revérifié le comportement du **template** de **sommaire** (**taxonomy-custom-categories.php**) pour m'assurer de la bonne association entre pages, catégories et pictogrammes. Cela m'a permis d'ajuster certaines incohérences d'affichages, notamment dans les cas où les catégories étaient associées à des pages sans avoir de sous-catégories.

### Affichage dynamique des actualités

L'intégration de la section « actualités » sur la page d'accueil s'est construite en collaboration. J'avais initialement mis en place la structure statique de cette section, destinée à accueillir trois cartes d'actualités. C'est ma tutrice qui a ensuite pris le relais pour y intégrer la dimension dynamique, en mettant en place la **requête WordPress** (**WP\_Query**) et la **boucle** correspondante, permettant d'afficher automatiquement les trois derniers **articles** publiés.

Elle a également ajouté le code **CSS** nécessaire pour styliser ces actualités sous forme de cartes, intégrant pour chacune le **thumbnail**, le titre, la date, ainsi qu'un lien cliquable redirigeant vers l'article complet.

Concernant la page **page-actualites.php**, j'ai de mon côté apporté plusieurs ajustements, notamment en retravaillant la mise en page et en y ajoutant du **CSS** via **TailwindCSS** pour fluidifier la lecture et l'organisation des éléments. Ces styles ont ensuite été affinés par ma tutrice, pour garantir une parfaite adéquation avec la charte graphique.

## Ajustements sur les blocs Gutenberg personnalisés

L'évolution des **blocs Gutenberg** a connu cette semaine une phase d'optimisation et de résolution de bugs. Le bloc **accordéon**, bien que fonctionnel lorsqu'il est inséré dans une colonne (grâce au système de **InnerBlocks**), a révélé une instabilité dès lors qu'il était utilisé hors conteneur structurant.

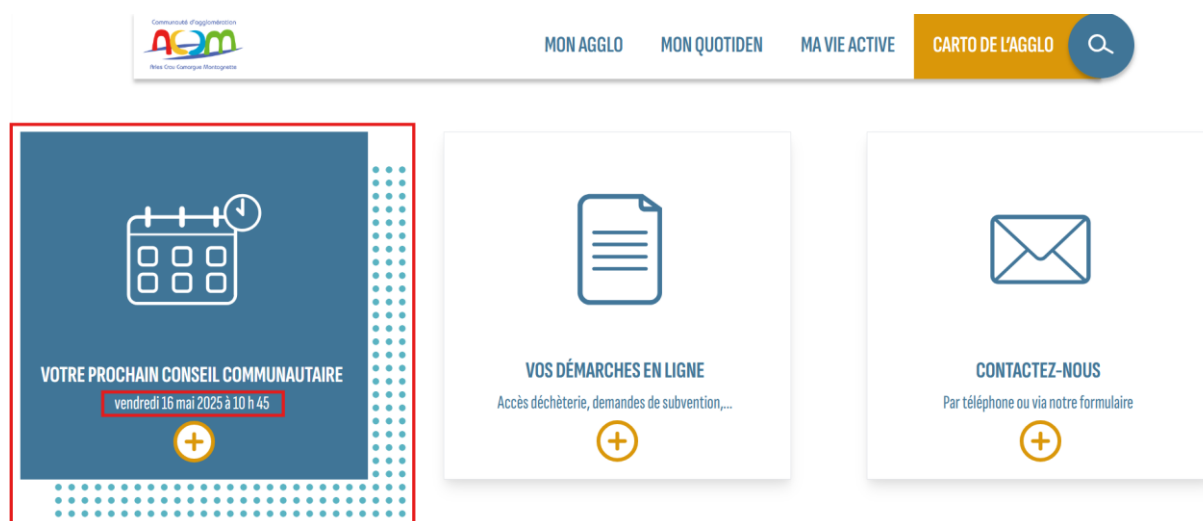
D'autres blocs ont connu des améliorations importantes :

**Bloc picto** : Créé pour permettre l'utilisation de la banque de pictogramme fournie par la designeuse. Ce bloc est désormais enrichi avec les mêmes options d'alignement que le bloc image natif de **WordPress**, grâce à l'intégration de **AlignmentToolbar** dans le **JSX**.

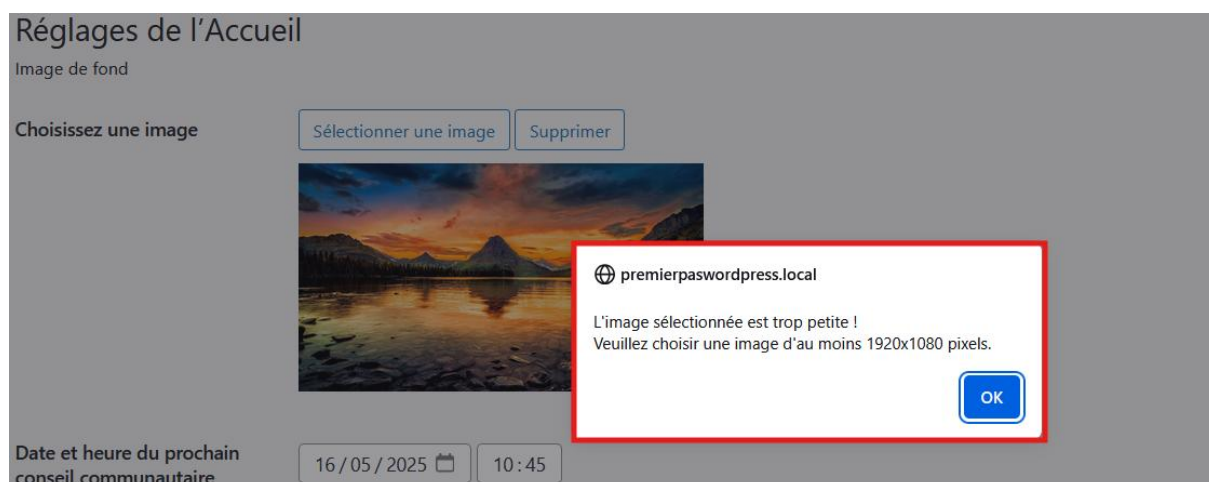
**Bloc couleur** : j'ai ajouté deux nouvelles couleurs (bleu et jaune), conformément aux codes graphiques de la charte ACCM. Un système d'alignement du contenu interne a également été introduit pour renforcer la flexibilité d'usage en front.

## Améliorations de la page d'accueil

Un nouveau champ personnalisé a été introduit dans le **CPT** « Réglage Accueil », permettant de saisir la date du prochain [Conseil Communautaire](#). Ce champ est ensuite affiché automatiquement sur la page d'accueil.



Pour garantir une qualité visuelle constante et sur demande des tuteurs, j'ai imposé une taille **minimale** d'image (1920x1080) pour les images de fonds de la page d'accueil. Cette décision a été motivée par l'existence d'un stock de photos professionnelles réalisées par un photographe de l'ACCM, et visait à éviter les images floues, mal cadrées ou pixélisées.



### Avancement sur les templates secondaires

Le développement des templates `404.php` et `search.php` a été amorcé. Ces pages, souvent négligées, jouent pourtant un rôle essentiel dans l'expérience utilisateur, surtout en cas d'erreur ou de recherche.

### Vie d'équipe et observations humaines

La semaine a également été rythmée par deux réunions importantes. La première, d'ordre institutionnel, a permis d'évoquer certains points liés à l'organisation générale du travail dans les services municipaux. Même si ce type de discussion dépasse le cadre strict de mon stage, elle m'a offert une meilleure compréhension du contexte dans lequel s'inscrivent les projets numériques. La seconde réunion, plus technique, s'est tenue avec le service Applications. Elle s'est déroulée dans un climat collaboratif, propice aux échanges, et m'a permis de clarifier plusieurs points techniques en vue des prochaines étapes du développement.

## Renforcement de la navigation et finalisation des composants dynamiques

Poursuivant l'évolution progressive du site, cette nouvelle semaine de développement a été écrite par un recentrage sur l'optimisation de la navigation, l'amélioration des blocs personnalisés, et la finalisation de certains templates.

### Affinage des templates de recherche et d'erreur

Dans une logique de confort utilisateur, j'ai apporté plusieurs ajustements aux templates secondaires du site. Le fichier `404.php` a été simplifié en réduisant le nombre d'articles suggérés de cinq à trois, pour alléger visuellement l'interface.

Le fichier `search.php`, quant à lui, a été entièrement révisé. Au début basé sur l'appel à des `template-parts`, il s'est avéré plus pertinent d'en revenir à une gestion directe dans le fichier principal, notamment pour corriger des erreurs de structure `HTML` et assurer une plus grande maîtrise du rendu. L'utilisation conditionnelle d'un `template-part` nommé `none` permet désormais d'informer clairement l'utilisateur en cas d'absence de résultats.

### Évolutions sur les blocs Gutenberg

Côté interface éditoriale, plusieurs ajustements ont encore été effectués sur les blocs personnalisés afin de les rendre plus cohérents avec les retours de l'équipe.

Le bloc Accordéon a été rendu plus flexible, autorisant l'insertion de tous types de blocs Gutenberg en son sein (j'avais autorisé que les blocs images ou paragraphes).

Le bloc Couleur a quant à lui été simplifié sur décision de ma tutrice. Les couleurs bleu et jaune ont été retirées, laissant place uniquement au rouge et au vert. Après des retours des utilisateurs, une largeur fixe a également été définie.

### Implémentation d'un système de menu dynamique

Un travail de fond a été mené sur la navigation principale du site. L'objectif était de remplacer le menu statique par une version dynamique, plus évolutive, et capable de refléter la structure réelle des contenus.

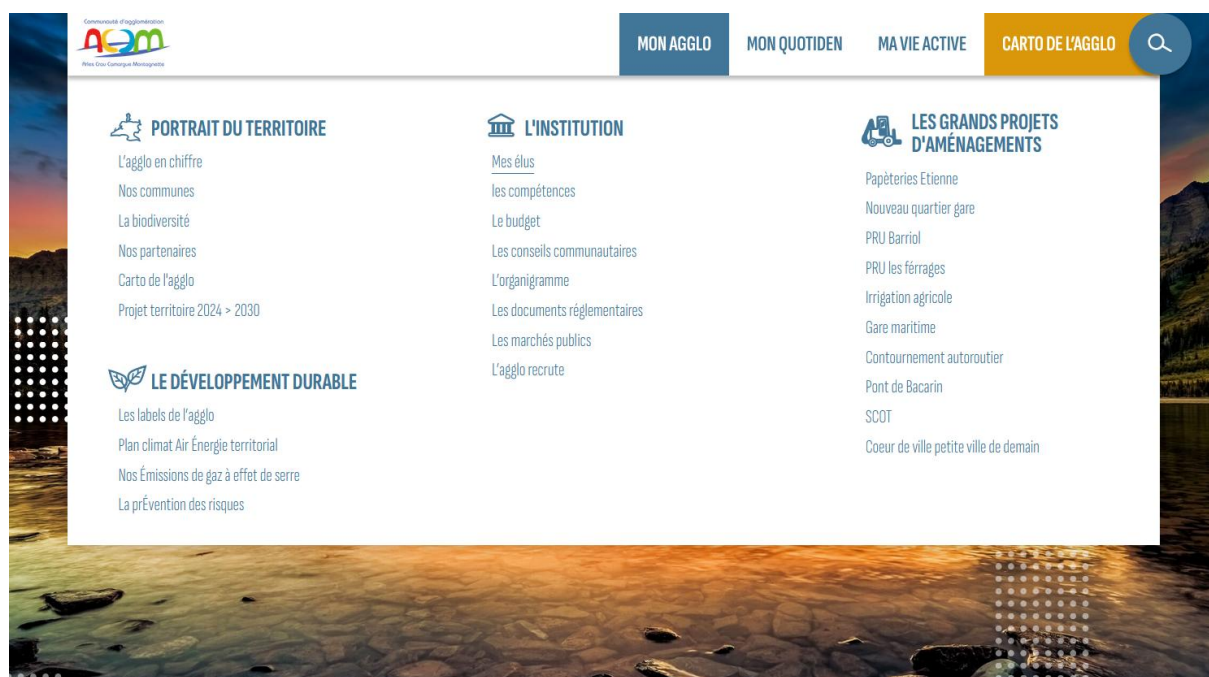
Ce développement m'a conduit à explorer certains concepts en **PHP**, notamment les **classes abstraites**, les **interfaces** et les **méthodes abstraites**, dans le but d'étendre correctement la classe **Walker\_Nav\_Menu**.

Un **Walker** personnalisé a ainsi été mis en place, permettant d'afficher uniquement les niveaux pertinents (catégories principales et sous-catégories de premier niveau), accompagnés de leurs pictogrammes respectifs.

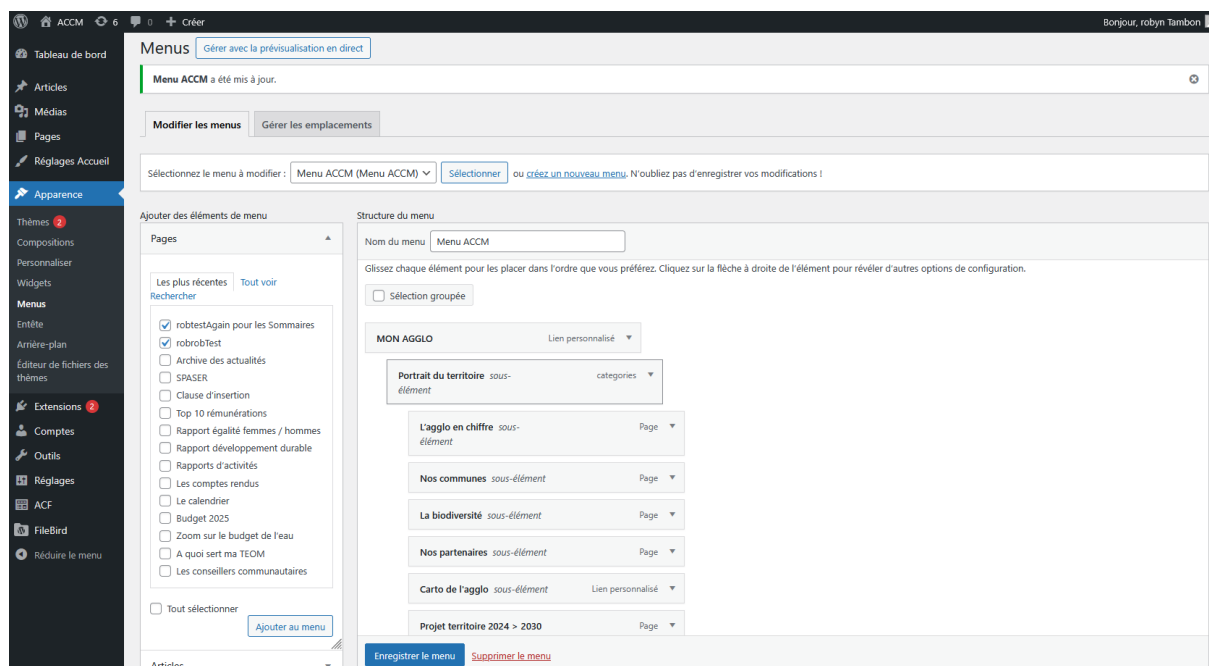
Ce système, injecté dans **header.php**, est complété par un **script JavaScript** permettant de gérer les événements utilisateurs sur les éléments de menu, pour une navigation plus fluide.

Cette fonctionnalité a nécessité une bonne partie de la semaine, tant les interactions entre les niveaux et les conditions d'affichage étaient nombreuses.

Rendu final du Header :



Et maintenant, les menus sont bien gérés via la fonctionnalité native de WordPress « Menus »



## Revue d'étape et retours

Le 10 avril, une présentation intermédiaire a été organisée afin de faire un point d'étape avec l'équipe. C'est moi qui ai été chargé d'animer cette présentation, ce qui m'a permis de mettre en valeur l'avancement de mon travail. Le fait que ma tutrice m'ait confié cette responsabilité témoigne de la confiance qu'elle m'accorde, malgré l'ampleur des ajustements encore nécessaires. Cette réunion a permis de recueillir plusieurs retours constructifs : amélioration du référencement (notamment via la balise <title>), suppression des dates sur les articles, et ajout de possibilités de personnalisation des différentes sections en dessous de la section des communes sur la page d'accueil pour répondre à des besoins politiques.

## Améliorations diverses

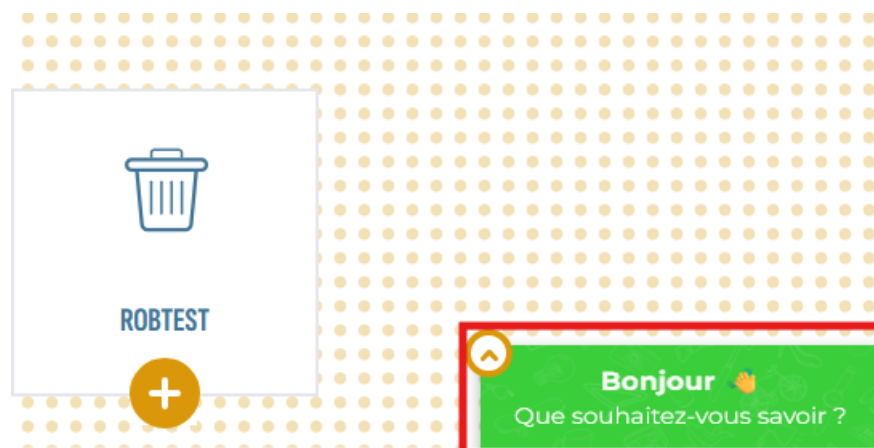
Plusieurs optimisations ont alors été réalisées :

Ajout d'un lien vers Google Maps dans le pied de page pour localiser l'agglomération.

Affichage **Trizzy** sur la catégorie « Mes Déchets » et toutes ses sous catégories, il s'agit d'un chabot existant déjà sur l'ancien site, il est affiché via un **iframe** plus du **CSS**, il a été par la suite séparé en **partials** et chargé dans le footer par ma tutrice.



On peut aussi le rabattre si on le souhaite

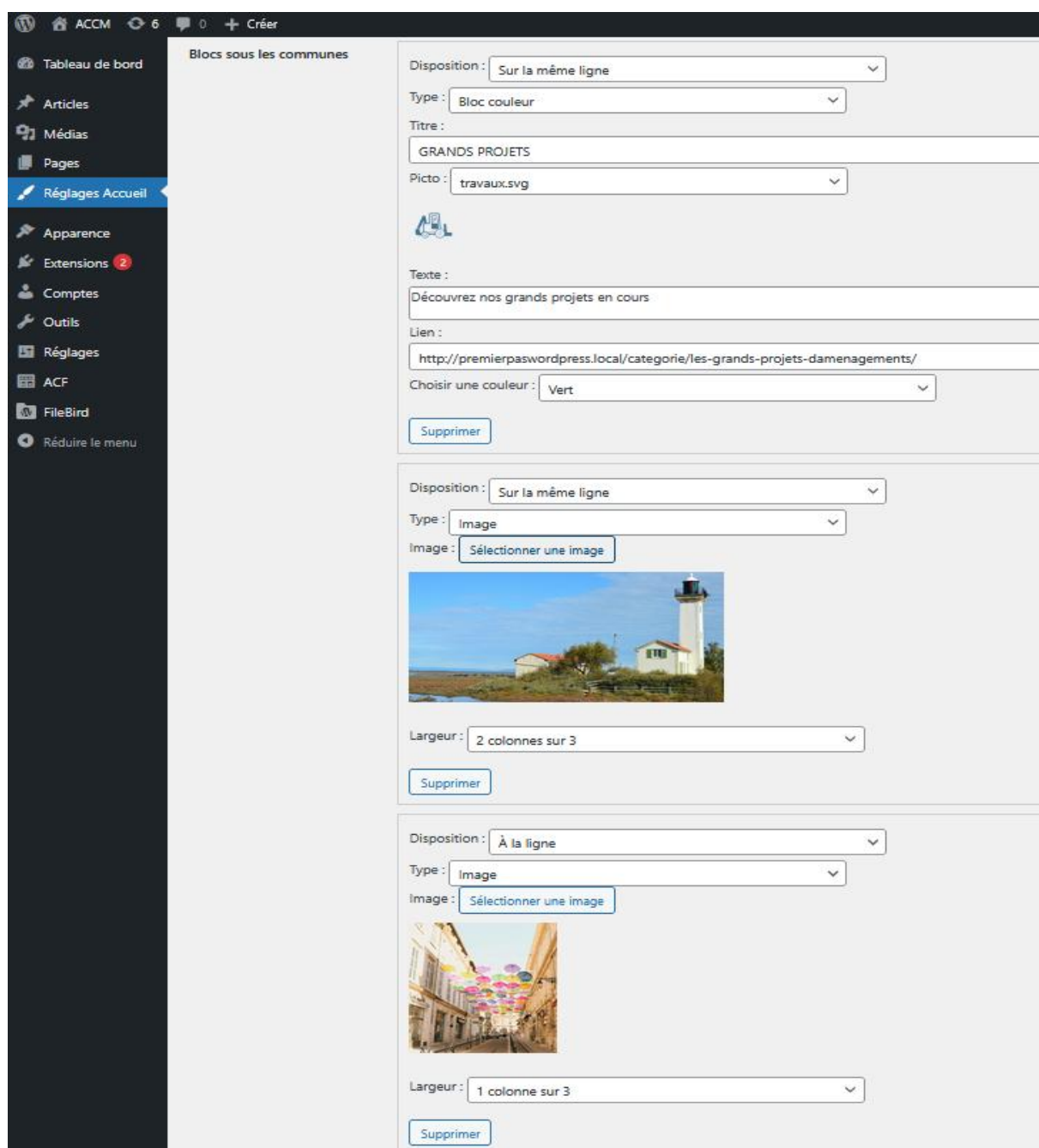


Modification de la balise `<title>` dans l'onglet des navigateurs, au format « ACCM - Nom de la page » dans une optique d'optimisation du **référencement naturel** ou **SEO**.

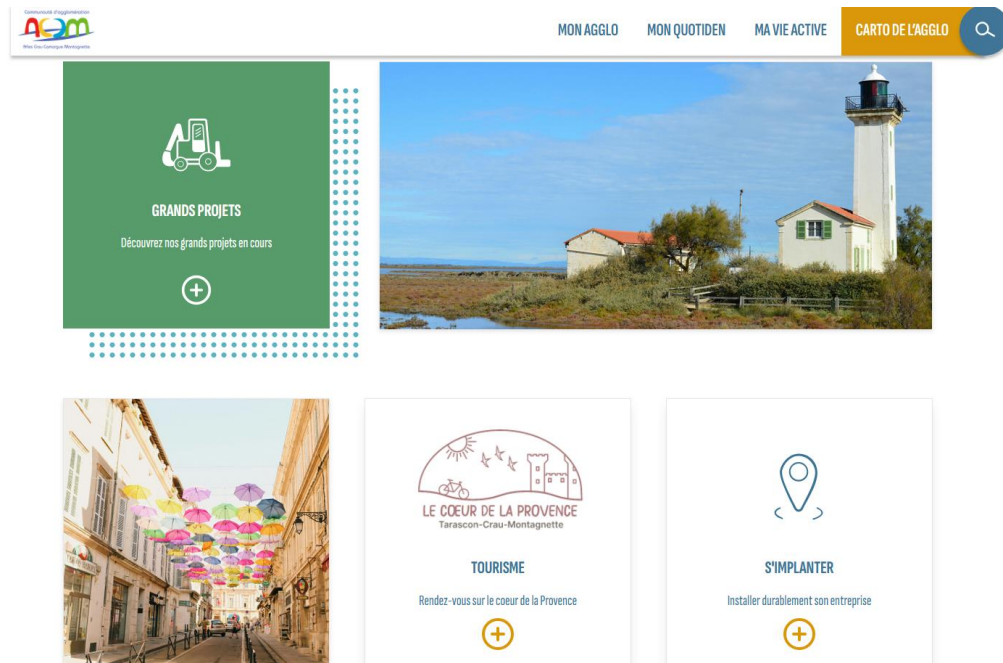
Mise à jour du style des titres en double : ils s'affichent désormais en bleu clair.

## Blocs dynamiques via la page de menu "Réglage Accueil"

Dans le but de donner davantage d'autonomie aux utilisateurs et de répondre à leurs demandes, j'ai poursuivi le développement de la page de menu « Réglage Accueil ». Celle-ci permet désormais de définir une suite de blocs dynamiques depuis le **back-office**.



Avec un affichage en front conditionné par leur ordre. Cela ouvre la voie à une gestion modulaire de la page d'accueil, en conservant une structure encadrée.



## Difficultés techniques rencontrées

Plusieurs défis ont été surmontés cette semaine :

La mise en place du menu dynamique a exigé une compréhension approfondie des **métadonnées** et de leur intégration en **JavaScript**.

Enfin, ma tutrice a exprimé une préférence pour une organisation **CSS** plus lisible, j'ai donc envisagé dans un premier temps de centraliser les classes utilitaires de **Tailwind** à l'aide de la directive **@apply**, dans l'objectif de faciliter la maintenance du projet à long terme.

## Vers une meilleure maintenabilité : de TailwindCSS à SASS

Après une phase de montée en puissance sur le développement fonctionnel, cette semaine a marqué un véritable tournant dans la stratégie technique du projet. Alors que le thème WordPress atteignait une stabilité appréciable, notamment grâce à la validation de la page de menu « Réglage Accueil » jugé fonctionnel et allant même au-delà des attentes initiales – une réorientation majeure m’a été imposée : **l’abandon complet de Tailwind CSS**.

### Abandon de TailwindCSS

Initialement, j’avais commencé à mettre en place la logique responsive du site en exploitant les breakpoints `sm`, `md`, `lg` de Tailwind, notamment sur l’affichage du menu en fonction des résolutions desktop, tablette et mobile. Ce travail préliminaire avait posé de bonnes bases, mais les tuteurs ont exprimé des réserves importantes sur la maintenabilité du projet à long terme avec Tailwind. En conséquence, la décision a été prise de supprimer TailwindCSS du projet. Même si cette nouvelle m’a demandé une réadaptation rapide, je l’ai abordée avec pragmatisme, j’ai amorcé une transition manuelle vers du CSS natif, avec une migration progressive des styles dans les fichiers `header.php`, `search.php` et les scripts JS associés. Un plan de migration vers SASS a également été défini pour structurer les feuilles de style de manière modulaire et maintenable.

### Retours utilisateurs et implémentation

Suite à une mini-formation que j’ai animée pour présenter le fonctionnement du back-office et du thème, la designeuse a mené une démonstration auprès de certains agents de l’ACCM après laquelle une liste précise d’ajustements visuels et ergonomiques m’a été communiquée. Parmi les demandes :

- Modifier le titre de la page d’accueil,
- Ajouter du padding sur certaines sections pour alléger l’affichage
- Transformer certains arrière-plans en pseudo-éléments `::before`
- Renommer les blocs personnalisés au format « Accm-NomDuBloc »
- Corriger les flèches dans le bloc Accordéon et fixer la largeur des images dans les blocs personnalisés
- Revoir le footer, jugé trop volumineux
- Repenser certains styles dans les taxonomies pour corriger des débordements
- Adapter les tailles et poids typographiques dans les en-têtes

Tous ces ajustements ont été progressivement intégrés dans le plan de transition. Ils témoignent de la finesse du regard graphique apporté au projet et m'ont permis d'affiner mon sens du détail.

### Gestion d'un bug ACF complexe

Un autre point critique a été la résolution d'un bug bloquant lié à **ACF**. Lors de la sauvegarde d'un terme, un message d'erreur bloquant apparaissait : '**mise à jour impossible pour footnotes**'. Ce champ fantôme, était absent du code mais présent dans la base de données. Après avoir identifié ce champ via une inspection **SQL**, j'ai procédé à un nettoyage manuel en base de données, ce qui a immédiatement rétabli un fonctionnement normal. Cette intervention m'a permis de renforcer ma compréhension de la structure des données **WordPress**, et de mieux appréhender l'impact des migrations de contenus.

```
DELETE FROM wp_postmeta  
WHERE meta_key IN ('footnotes', '_footnotes');
```

### Finalisation de la suppression de Tailwind

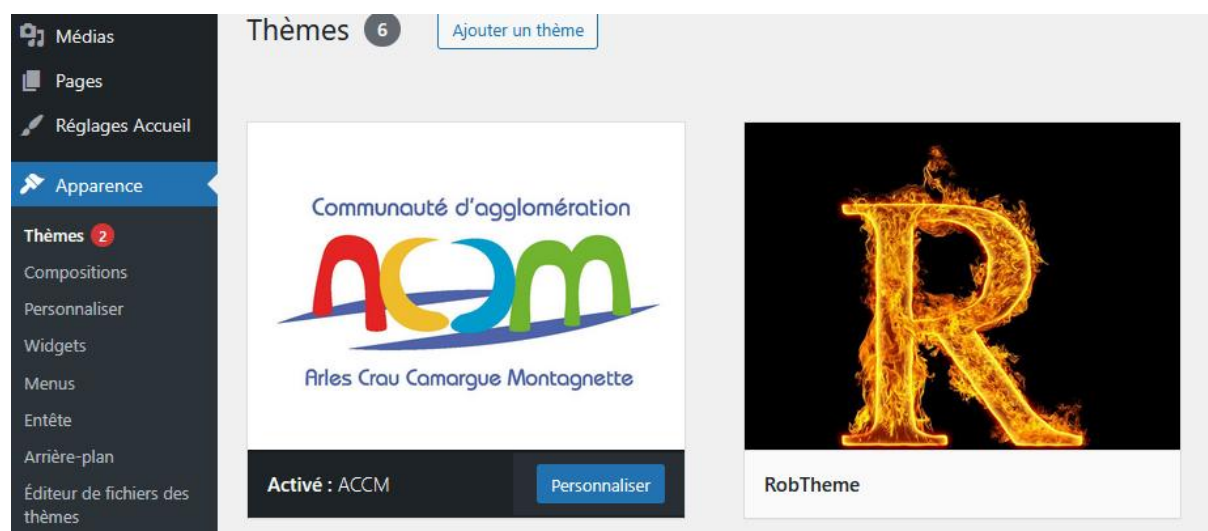
La semaine a été ponctuée par la poursuite du retrait de **TailwindCSS**, amorcé les jours précédents, et la transition vers une architecture en **CSS** pur. Ce changement s'est accompagné de la mise en place d'un environnement **Sass**.

L'ensemble des classes utilitaires utilisées jusque-là dans les fichiers **PHP**, **JS** et **JSX** a été remplacé méthodiquement, ligne par ligne, par des **sélecteurs CSS** personnalisés. Cette démarche, a nécessité la modification manuelle et vérification de plus de quarante fichiers répartis dans les dossiers du **thème WordPress** : les templates principaux (**header.php**, **footer.php**, **front-page.php**, etc.), les fichiers de structure (**404.php**, **category.php**, **search.php**, etc.) ainsi que les **scripts JavaScript** tels que **menu-deroulant.js**, initialement pensés avec les logiques de **Tailwind**. Chaque fichier a été relu et corrigé avec minutie pour assurer la continuité visuelle et fonctionnelle du site.

Les ajustements concernaient aussi bien les effets de survol, les marges, les typographies, que la gestion des couleurs ou des pictogrammes. La moindre erreur de conversion pouvait dégrader l'interface, et j'ai donc dû valider chaque changement par des tests visuels. Certaines sections comme `front-page.php` ou `taxonomy-custom-categories.php` ont été particulièrement sensibles, nécessitant plusieurs allers-retours pour retrouver une cohérence avec la maquette d'origine.

À l'issue de cette longue phase de remplacement, `TailwindCSS` a été entièrement retiré du projet, y compris son fichier `output.css`, remplacé par une nouvelle chaîne de compilation `Sass`. J'ai configuré l'environnement en installant les dépendances nécessaires et en convertissant les anciens fichiers `.css` en `.scss`.

Une étape importante a consisté à générer un fichier `style.css` contenant l'en-tête du thème `WordPress`. Cet en-tête est un commentaire placé en haut du fichier principal des styles, qui décrit les informations essentielles du thème (nom, auteur, version, etc.). Il est indispensable, car sans lui, `WordPress` ne reconnaît pas le thème dans l'interface d'administration. Sur recommandation de Frederic Cases, j'ai donc déplacé cet en-tête dans le fichier `input.scss` afin qu'il soit conservé après compilation, dans le `style.css` généré automatiquement à la racine du thème.



Cette migration ne s'est pas faite sans douleur. Dès les premières compilations, le linter `Sass` a remonté plus de 2000 erreurs, liées à des problèmes d'indentation, d'espacements autour des sélecteurs (`+`, `>`, etc.), des unités, des pseudo-éléments, ou encore à cause de l'absence de lignes

finale dans certains fichiers. Une phase de refactorisation complète du code **Sass** a donc été nécessaire.

Par-dessus cela, j'ai dû corriger une nouvelle fois des erreurs d'affichage pourtant déjà résolues avant la transition. Certaines mises en page, parfaitement fonctionnelles auparavant, avaient perdu leur logique graphique : paddings erronés, hiérarchie des containers déstructurée, incohérences dans les titres ou les espacements. Cette étape s'est révélée particulièrement fastidieuse : j'ai dû faire, refaire, et re refaire plusieurs composants, en revenant sur des ajustements déjà validés. Malgré la frustration, cette phase m'a permis de renforcer ma rigueur et de mieux comprendre l'impact d'un changement d'architecture sur un projet avancé.

Ce travail de nettoyage final a abouti à un résultat clair et conforme :

Le site repose désormais sur une base 100 % **Sass**.

Plus aucune dépendance à **TailwindCSS** n'existe.

Le fichier style.css est proprement généré à la racine du thème, conformément à l'exigence de Frederic Cases, et contient l'en-tête nécessaire à la reconnaissance du thème dans le back-office **WordPress**.

D'autre part, j'ai mis en place plusieurs outils afin de pérenniser la qualité du code :

**Stylelint** a été intégré pour forcer une rigueur dans la syntaxe CSS/SCSS (espacements, indentation, conventions).

**RTLCS** a été configuré, dans la perspective d'une future internationalisation du site (prise en charge des langues s'écrivant de droite à gauche).

Le package `@wordpress/scripts` a été ajouté pour garantir la conformité aux standards **WordPress**, notamment pour **Babel**, le **linter JS**, et la **compilation SCSS**.

Un petit utilitaire, **Dir-archiver**, a aussi été ajouté pour générer une archive propre du thème à des fins de livraison, même s'il n'est pas encore utilisé.

Mon environnement de travail a lui aussi été harmonisé :

J'ai modifié les préférences de Visual Studio Code pour ajouter automatiquement une ligne vide en fin de fichier (`file.insertFinalNewline: true`).

J'ai uniformisé les fins de lignes en passant du format **CRLF** (Windows : `\r\n`) **LF** (Unix : `\n`) *Line Feed*, ce qui permet d'éviter des conflits d'encodage avec les environnements **Linux**.

Une fois la base **Sass** stabilisée, j'ai pu m'attaquer aux retours remontés par les utilisateurs et la designeuse :

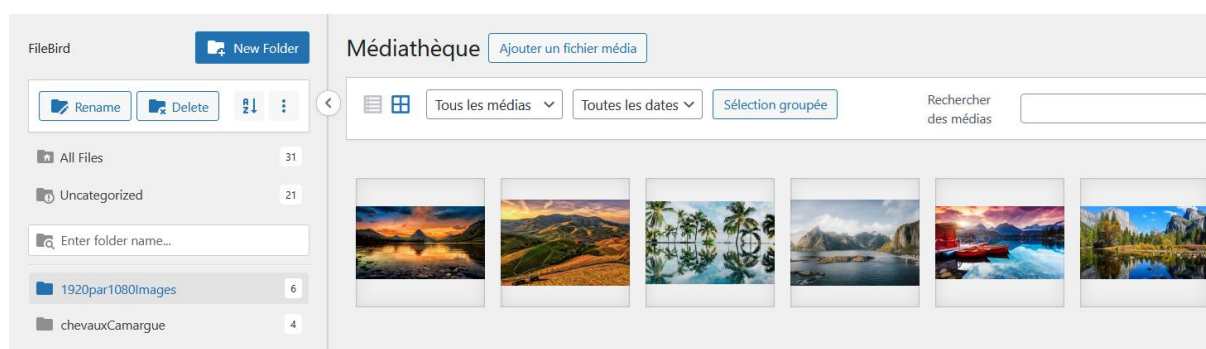
Le container global a été redéfini selon les spécifications Figma, avec des largeurs progressives selon les **breakpoints** (de 100 % à 1320px).

	Extra small	Small	Medium	Large	X-Large	XX-Large
	<576px	≥576px	≥768px	≥992px	≥1200px	≥1400px
container	100%	540px	720px	960px	1140px	1320px

Les titres des menus ont également été revus pour coller aux tailles exactes définies dans les composants graphiques.

D'autres corrections ciblées ont été apportées, notamment au niveau des espacements.

J'ai aussi répondu à une demande d'organisation de la **médiathèque WordPress**. Étant donné que **WordPress** ne propose pas nativement d'arborescence de fichiers, j'ai effectué une recherche comparative pour identifier un plugin adapté. **FilesBird** a été installé à titre provisoire, permettant aux utilisateurs de créer des dossiers visuels dans la médiathèque. Cet outil, permet d'ajouter une surcouche visuelle à la **médiathèque**, sous forme de dossiers dans lesquels on peut glisser-déposer les fichiers, un peu à la manière de l'explorateur **Windows**. Il ne modifie ni les **URL**, ni les fichiers, et n'a aucun impact sur le site public ou ses performances, ce qui en faisait une solution légère et temporaire adaptée au contexte du projet. Développer un système similaire en interne aurait nécessité du temps difficilement justifiable à ce stade, parce que la phase de **responsive design** n'était pas encore entamée, il a finalement été supprimé.



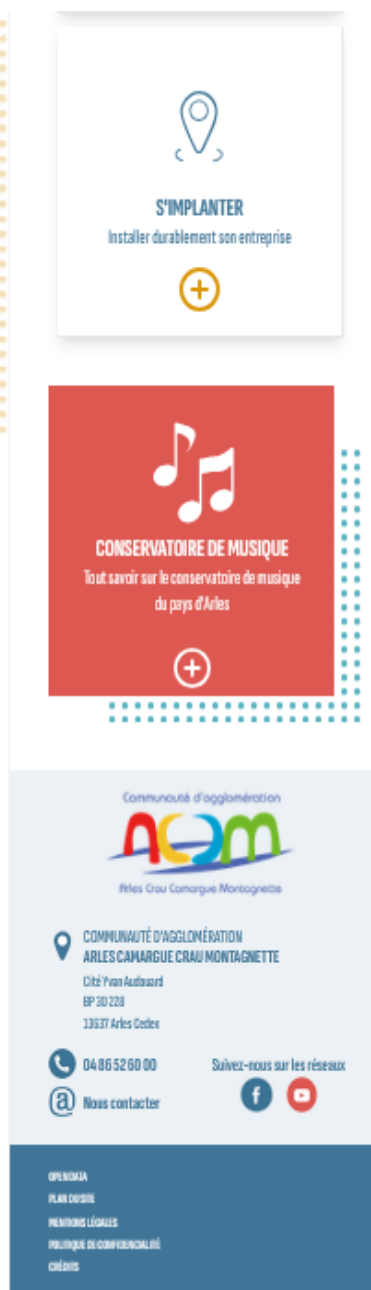
Enfin, le responsive mobile a été lancé le 30 avril, en commençant par la page d'accueil. J'ai intégré un système de swipe horizontal pour les blocs "Accès rapide" et les actualités, La page d'accueil a été finalisée en mobile dès le vendredi 2 mai. Voici un aperçu en trois colonnes conforme à la maquette mobile :



Haut de la page



Milieu de la page



Bas de la page

Pour garantir la fraîcheur du contenu lors des mises à jour et sur demande de Frederic Cases, j'ai aussi intégré une logique de **cache busting**. Grâce à `filemtime()` dans la fonction `wp_enqueue_style()`, chaque changement de **style.css** et de tous les **scripts** génère une nouvelle version, forçant le navigateur à recharger les fichiers à jour.

## Décision institutionnelle et report du projet

Le 7 mai 2025, un message officiel a annoncé un changement majeur concernant le calendrier de mise en ligne du site. Au départ prévu pour le 4 juillet, la publication du site a été repoussée à l'année 2026, à l'issue des prochaines élections municipales. Cette décision, prise par Pierrick Pouget, vise à garantir que l'outil mis à disposition des usagers soit entièrement finalisé, complet, et pleinement fonctionnel, sans compromis liés à des délais serrés.

Ce report change profondément la perspective du projet :

La nouvelle échéance est fixée à 2026, offrant un délai supplémentaire pour peaufiner, tester et enrichir les fonctionnalités déjà mises en place.

Sur le plan personnel, cette annonce a été un peu difficile à encaisser. J'aurais évidemment aimé voir en ligne le fruit de mon travail, après des semaines d'implication technique et créative. Malgré tout, je comprends la logique institutionnelle derrière cette décision, et j'espère sincèrement que le site sera valorisé et déployé comme prévu, car il représente un véritable investissement de ma part.

## Améliorations du back-office, tri personnalisé et lancement de la rédaction

En cette treizième semaine de stage, mon travail a été fortement orienté vers deux axes : l'amélioration de l'expérience utilisateur dans la navigation du site (à travers la pagination et un système de tri sur mesure) et la préparation de la rédaction de ce présent mémoire, rendue possible grâce au temps dégagé suite au report du déploiement.

## Mise en place de la pagination native WordPress

Afin d'améliorer l'ergonomie de la page d'actualités, d'en alléger la lecture et sur demande de ma tutrice, j'ai implémenté une pagination native inspirée du fonctionnement du site arles.fr. Le système repose sur la variable `$paged`, permettant de détecter dynamiquement la page en cours dans une requête personnalisée. J'ai ensuite intégré la fonction `paginate_links()` pour générer le **HTML** de navigation, avec une personnalisation visuelle des flèches "précédent" et "suivant". L'ancien affichage brut des actualités a ainsi été remplacé par une interface plus claire et plus fonctionnelle.

## Développement d'une fonctionnalité de tri personnalisée pour les pages sommaires

L'un des objectifs fixés lors des réunions précédentes était de permettre un tri manuel des sous-catégories et sous-pages dans les pages sommaires. Pour cela, j'ai ajouté un champ **ACF** sortable à l'interface d'édition des pages et des taxonomies, couplé à une fonction de sauvegarde des données triées.

Cette fonctionnalité a nécessité de nombreux ajustements techniques :

Mauvaise injection du champ dans le formulaire ACF, due à la gestion asynchrone via AJAX → corrigée avec le **hook JS** `acf.addAction('submit')`.

Tableau de tri vide dans `acf/save_post` → la méthode de récupération a été remplacée par une approche plus fiable en **jQuery**.

Ordre non sauvegardé côté PHP → solutionnée via la création manuelle d'un tableau trié lors de l'affichage.

Conflit de détection `$screen->post_type` dans l'édition de termes → remplacé par `$screen->base` pour plus de précision et `term_id` toujours à 0 dans certains cas → récupéré manuellement via `$_GET['tag_ID']`.

Une fois la structure stabilisée, la sauvegarde des métadonnées a été sécurisée, l'affichage en back-office fiabilisé, et une boucle PHP dynamique permet désormais d'afficher les contenus triés dans le front-end selon les préférences définies dans l'administration.

## Amélioration de l'expérience utilisateur en back-office

Pour éviter toute confusion côté éditeurs, j'ai affiné le comportement du champ de tri en utilisant le `hook acf/prepare_field`, afin qu'il n'apparaisse plus lors de la création d'un nouveau terme, mais uniquement en modification, où le tri est pertinent.

## Résolution d'un bug dans "Réglage Accueil"

Un bug affectait l'affichage des blocs sous les communes dans le menu "Réglage Accueil" : ceux-ci avaient disparu du back-office. Après investigation, le problème venait d'un script JS que j'avais temporairement commenté pour contourner une erreur 404 liée à un fichier inexistant. La solution a été de créer un script "virtuel" via `wp_register_script('admin-scripts', false)`, ce qui supprime l'erreur tout en conservant la variable JS nécessaire à l'injection des blocs.

## Mise en pause officielle du projet côté service Application

À ce stade du développement, et malgré l'avancement fonctionnel significatif, le projet a été officiellement mis de côté par le service Application. Cette décision fait suite à la communication institutionnelle reçue la semaine précédente.

Cette mise en veille du projet a modifié la dynamique interne : les priorités de l'équipe se sont recentrées sur d'autres chantiers techniques jugés plus urgents à court terme.

## Conclusion

Ce stage de troisième année, passé au sein de la DSI de la Mairie d'Arles, a été plus qu'une expérience professionnelle il a représenté pour moi une immersion dans un environnement institutionnel exigeant et humainement riche. J'ai pu évoluer dans un cadre où la rigueur technique et l'esprit d'équipe ne manquaient pas.

Le projet de refonte du site internet de l'Agglomération ACCM a représenté un défi à plusieurs niveaux. D'abord technique, avec la nécessité de créer un thème WordPress sur mesure, modulaire, administrable, accessible et fidèle à une maquette validée. Ensuite méthodologique, car il fallait s'adapter à des attentes évolutives, à des décisions stratégiques prises à plusieurs niveaux, et à des outils parfois complexes à maîtriser, comme Advanced Custom Fields, Gutenberg ou encore l'environnement SCSS. Enfin, humain, puisque ce travail s'est inscrit dans une logique collaborative constante, entre développeurs, graphistes, agents du service communication.

Tout au long du stage, j'ai su mobiliser mes compétences acquises à l'IUT et pendant mon BTS, les faire évoluer, les approfondir, et en découvrir de nouvelles. J'ai dû apprendre à remettre en question mes méthodes, notamment lors du changement de structure de contenu et lors du retrait progressif de TailwindCSS. Sur le plan humain, le stage m'a offert l'opportunité de développer des qualités essentielles comme la communication, la patience, mais aussi la capacité à écouter, et à travailler avec des profils aux attentes variées. Le fait que ma tutrice m'ait confié les présentations du projet devant l'équipe a été pour moi un signe fort de la confiance qu'elle m'accordait.

En conclusion, ce stage a pleinement répondu à mes objectifs initiaux : progresser en développement web, être plutôt à l'aise en PHP, comprendre les rouages des collectivités territoriales car ce stage fait suite à un précédent qui s'est déroulé à la mairie d'Aubenas, m'approprier les bonnes pratiques du développement WordPress, et surtout, gagner en autonomie et en professionnalisme. Il m'a également permis de mesurer l'importance du travail bien fait, même lorsqu'il n'est pas immédiatement visible en ligne. Ce projet ne se termine pas avec la fin de mon stage ; il se poursuit dans la mémoire technique de l'équipe, prêt à être réactivé. Et moi, j'en ressors grandi, mieux préparé aux exigences du monde professionnel, et plus confiant dans ma capacité à y trouver ma place.

## Résumé

Ce stage de 15 semaines que j'ai passé au sein de la Direction des Systèmes d'Information de la Mairie d'Arles, m'a permis de participer activement à la refonte complète du site internet de l'Agglomération ACCM. L'objectif était dès le départ de repartir de zéro, sur WordPress, en respectant une maquette graphique, tout en produisant un thème entièrement personnalisé et facile à maintenir.

Tout au long du projet, j'ai mis en place un environnement de développement structuré, appris à manier des outils avancés comme Sass, ACF, Webpack ou encore Babel, et conçu plusieurs blocs Gutenberg dynamiques. J'ai aussi dû faire face à des changements de cap importants, comme l'abandon de TailwindCSS en cours de route, ou encore à des problématiques complexes liées aux URLs, au tri des contenus, ou à l'ergonomie du back-office.

Au-delà des compétences techniques, ce stage m'a surtout appris à travailler dans un cadre institutionnel, à collaborer avec des profils variés (développeurs, communicants, graphistes...), à défendre mes choix tout en restant à l'écoute, et à tenir un cap malgré les imprévus. Même si la mise en ligne du site a été reportée à 2026, ce projet représente pour moi une vraie réussite, autant sur le plan professionnel que personnel.

## Mots-clés

Technologies et langages : WordPress ~ PHP ~ JavaScript / JSX ~ HTML / CSS ~ SASS / SCSS ~ TailwindCSS ~ Babel ~ Webpack ~ WP-CLI.

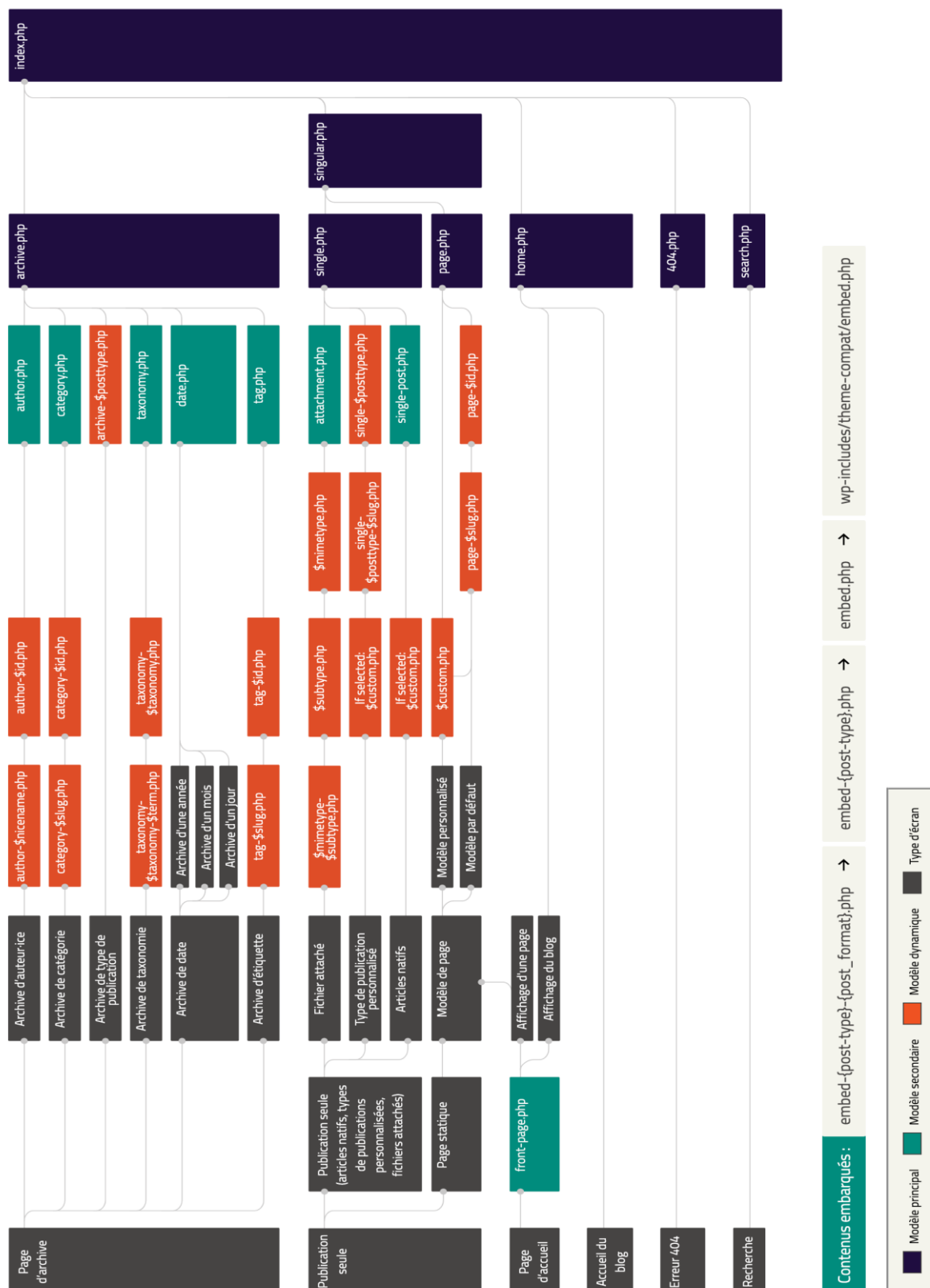
Fonctionnalités WordPress : CMS ~ Thème sur-mesure ~ Gutenberg ~ Blocs personnalisés ~ Advanced Custom Fields ~ Custom Post Type ~ Taxonomies personnalisées ~ Menu dynamique ~ Back-office / Front-end.

Design & UX : Responsive Design ~ Charte graphique ~ Figma ~ Pictogrammes.

Contexte professionnel : DSI (Direction des Systèmes d'Information) ~ Service Applications Collectivité ~ ACCM (Arles Crau Camargue Montagnette) ~ Site institutionnel.

# Annexes techniques

## Structure des templates WordPress



## Arborescence du thème WordPress ACCM

accm-theme/

- blocks/	→ Blocs Gutenberg personnalisés
- accordion/	
- accordion/	
- accordion-info/	
- bloc-couleur/	
- picto/	
- accordion.php	
- accordion.php	
- accordion-info.php	
- bloc-couleur.php	
- picto.php	
- dist/	→ Fichiers CSS compilés par TailwindCSS (supprimé)
- img/	→ Ressources graphiques
- inc/	→ Fichiers utilitaires et fonctions PHP
- js/	→ Scripts JavaScript spécifiques
- node_modules/	→ Modules NPM
- partials/	→ Morceaux de templates HTML/PHP
- sass/	→ Fichiers sources SASS (SCSS)
- .babelrc	→ Configuration Babel
- .editorconfig	→ Configuration d'éditeur
- .eslintrc	→ Linter JS
- .stylelintrc.json	→ Linter CSS/SASS
- 404.php	→ Template de page introuvable
- category.php	→ Template pour les catégories d'articles
- composer.json	→ Dépendances PHP (si utilisées)
- fil-ariane.php	→ Composant du fil d'Ariane
- footer.php	→ Pied de page du site
- front-page.php	→ Template de la page d'accueil
- functions.php	→ Fonctions globales du thème
- header.php	→ En-tête du site
- index.php	→ Point d'entrée du thème
- package.json / package-lock.json	→ Dépendances Node.js
- page.php	→ Template générique de page
- page-actualites.php	→ Page d'actualités
- page-archive-actualites.php	→ Archives des actualités
- readme.md	→ Documentation interne
- screenshot.jpg	→ Capture d'écran du thème
- search.php	→ Template des résultats de recherche
- style.css	→ Feuille de style principale générée à partir de SCSS
- taxonomy-custom-categories.php	→ Affichage des taxonomies personnalisées
- webpack.config.js	→ Configuration Webpack (compilation JS)

## Feuille de temps

	Dates	Tâches	Heures
1	17/02/25 - 21/02/25	Mise en place de l'environnement local, installation de WordPress, début apprentissage TailwindCSS + WordPress	35h
2	24/02/25 - 28/02/25	Structuration initiale du thème, début intégration du header/footer, configuration de Tailwind	35h
3	03/03/25 - 07/03/25	Découverte ACF, modularisation (template parts), champs personnalisés	35h
4	10/03/25 - 14/03/25	CPT + taxonomies, front-page, développement de single.php, logique d'arborescence	35h
5	17/03/25 - 21/03/25	Restructuration des contenus, création blocs Gutenberg (bloc-couleur, accordéon), configuration Webpack/Babel	35h
6	24/03/25 - 28/03/25	Finalisation blocs, adaptation front-page, sommaire taxonomie, ajout bouton "Back to top", page actualités	35h
7	31/03/25 - 04/04/25	Ajout contenu (pages, catégories), affichage actualités, ajustements des blocs	35h
8	07/04/25 - 11/04/25	Walker menu personnalisé, refonte menu mobile, responsive partiel, début conversion CSS pur	35h
9	14/04/25 - 18/04/25	Abandon Tailwind, migration CSS ligne par ligne, retours utilisateurs/design	35h
10	21/04/25 - 25/04/25	Suppression totale de Tailwind, installation Sass, refacto de +40 fichiers, intégration RTLCSS et Stylelint	28h lundi férié
11	28/04/25 - 02/05/25	Finalisation passage à Sass, responsive mobile page d'accueil, bug fixes, organisation médias	28h jeudi férié
12	05/05/25 - 09/05/25	Correction URLs et slugs, adaptation mobile.	28h jeudi férié
13	12/05/25 - 16/05/25	Pagination actualités, tri sous-catégories/pages, amélioration interface ACF	35h
14	19/05/25 - 23/05/25	Rédaction mémoire	35h
15	26/05/25 - 30/05/25	Rédaction mémoire	28h jeudi férié

## Glossaire

**ACCM** : Arles Crau Camargue Montagnette

**ACF** : Advanced Custom Fields

**CMS** : Content Management System

**CPT** : Custom Post Type

**CRLF** : Carriage Return + Line Feed (\r\n)

**CSS** : Cascading Style Sheets

**HTML** : HyperText Markup Language

**JSX** : JavaScript XML

**LF** : Line Feed

**PHP** : Hypertext Preprocessor

**RTL CSS** : Right To Left Cascading Style Sheets

**SASS** : Syntactically Awesome Style Sheets

**SCSS** : Sassy Cascading Style Sheets

**SEO** : Search Engine Optimization

**SQL** : Structured Query Language

**SVG** : Scalable Vector Graphics

**URL** : Uniform Resource Locator

## Lexique

**Advanced Custom Fields** : Plugin WordPress permettant de créer et gérer des champs personnalisés pour enrichir les contenus.

**Articles WordPress** : Contenus publiés régulièrement.

**Babel** : Outil JavaScript qui rend du code moderne compatible avec les anciens navigateurs.

**Back-office WordPress** : Interface d'administration utilisée pour gérer le site.

**Blocs Gutenberg** : Composants modulaires de l'éditeur WordPress permettant de construire des mises en page flexibles.

**Breackpoints Tailwind** : Points de rupture définis dans TailwindCSS pour gérer le responsive design selon les tailles d'écran.

**Cache busting** : Technique utilisée pour forcer les navigateurs à recharger des fichiers modifiés.

**Custom Post Type** : Type de contenu personnalisé créé dans WordPress en plus des pages et articles standards.

**Dir-archiver** : Outil permettant de compresser un répertoire entier dans un fichier zip.

**En tête du thème WordPress** : Partie du fichier style.css contenant les informations principales du thème (nom, auteur, etc.).

**Fil d'Ariane** : Navigation secondaire qui montre le chemin hiérarchique jusqu'à la page actuelle.

**FilesBird** : Plugin WordPress permettant de mieux organiser la médiathèque en créant des dossiers virtuels.

**Gutenberg** : Éditeur visuel de WordPress basé sur des blocs.

**Hooks** : Points d'ancrage dans WordPress permettant d'exécuter du code à des moments précis.

**InnerBlocks** : Composant Gutenberg permettant d'insérer des blocs enfants dans un bloc parent personnalisé.

**JavaScript** : Langage de programmation côté client utilisé pour rendre les sites interactifs et dynamiques.

**JQuery** : Bibliothèque JavaScript simplifiant la manipulation du DOM, les événements et les requêtes AJAX.

**Linter Sass** : Outil d'analyse qui vérifie le code Sass pour respecter des règles de style et de bonnes pratiques.

**Local** : Environnement de développement local pour WordPress, développé par Flywheel.

**Logiciels libres** : Programmes distribués avec leur code source, modifiables et redistribuables librement.

**Médiathèque WordPress** : Espace de stockage des fichiers médias (images, vidéos, documents) dans le tableau de bord.

**Metadonnées** : Données supplémentaires associées à un contenu.

**MODX** : CMS open source, permettant une gestion de contenu.

**MySQL** : Système de gestion de base de données relationnelle.

**Nginx** : Serveur web performant.

**OpenAdminerEvo** : Outil d'administration de base de données.

**Page de menu** : Écran personnalisé ajouté dans l'administration WordPress pour accéder à une interface spécifique liée à un thème ou un plugin.

**Plugin** : Extension logicielle permettant d'ajouter des fonctionnalités à WordPress sans le modifier.

**PostCSS** : Outil de transformation CSS permettant d'utiliser des plugins pour ajouter des fonctionnalités avancées.

**React** : Bibliothèque JavaScript développée par Meta, utilisée pour créer des interfaces utilisateurs dynamiques (notamment Gutenberg).

**Référencement naturel** : Ensemble de techniques visant à améliorer la visibilité d'un site sur les moteurs de recherche (SEO).

**Requête WordPress** : Processus interne qui détermine quel contenu afficher selon l'URL ou les paramètres passés.

**RTLCSS** : Outil qui convertit les fichiers CSS pour prendre en charge les langues s'écrivant de droite à gauche.

**Script** : Fichier JavaScript chargé dans une page pour ajouter des comportements interactifs.

**Sélecteurs CSS** : Expressions utilisées en CSS pour cibler des éléments HTML et leur appliquer des styles.

**Slug** : Partie de l'URL qui identifie un contenu de manière lisible.

**TailwindCSS** : Framework CSS utilitaire qui permet de construire des interfaces rapidement via des classes prédéfinies.

**Taxonomie** : Système de classification des contenus dans WordPress (ex : catégories).

**Templates** : Fichiers PHP définissant la structure des pages.

**Terme de taxonomie** : Élément spécifique d'une taxonomie.

**The Loop** : Boucle principale de WordPress qui permet d'afficher dynamiquement les contenus selon la requête.

**Thumbnail** : Image miniature associée à un article ou un contenu, et affichée dans les aperçus.

**Theme WordPress** : Ensemble de fichiers qui contrôlent l'apparence visuelle et la structure d'un site WordPress.

**Trizzy** : Assistant numérique local destiné à guider les utilisateurs sur le tri des déchets.

**UX** : Expérience utilisateur, désigne la qualité de l'interaction entre un utilisateur et le site.

**Walker** : Classe PHP permettant de personnaliser l'affichage des menus ou listes générés dynamiquement.

**Webpack** : Outil JavaScript qui regroupe, transforme et optimise les fichiers (JS, CSS) en un seul ensemble prêt pour la mise en production.

**WP-CLI** : Interface en ligne de commande pour gérer un site WordPress sans passer par l'interface graphique.

**WP\_Error** : Classe WordPress utilisée pour gérer les erreurs lors des appels de fonctions ou d'API.

# Bibliographie

## Ressources textuelles

**Banque des Territoires.** (2018). *Arles : des logiciels libres pour des applications métiers sur mesure (13)*. Banque des Territoires. Consulté le 19 mai 2025 sur :

<https://www.banquedesterritoires.fr/experience/arles-des-logiciels-libres-pour-des-applications-metiers-sur-mesure-13>

**La Gazette des Communes.** (2011). *La mairie d'Arles, pionnière du logiciel libre*. Consulté le 19 mai 2025, sur : [https://adullact.org/images/sampled/Presse/article\\_Gazette\\_26092011.pdf](https://adullact.org/images/sampled/Presse/article_Gazette_26092011.pdf)

**WordPress.org.** (2025). *Documentation officielle de WordPress*. Consulté tout au long du stage sur : <https://wordpress.org/documentation>

Autres liens utilisés :

<https://arles.fr/>

<https://www.agglo-accm.fr/>

<https://annuaire-entreprises.data.gouv.fr/entreprise/commune-d-arles-211300041>

## Ressources vidéo

**DYWANTS.** (2023). *Comment développer son thème WordPress sur mesure ?* Consulté le 18 février 2025 sur : <https://www.youtube.com/watch?v=ki2cHnKIg5E>

**Grafikart.fr.** (2019). *Tutoriel WordPress : Créer un bloc Gutenberg*. Consulté le 5 mars 2025, Sur : <https://grafikart.fr/tutoriels/gutenberg-block-wordpress-1145> ainsi que ses séries de vidéos « Créer un thème WordPress » et « Apprendre React ».

Et divers vidéos de **wpMarmite** sur <https://www.youtube.com/@WPMarmite>