WordPress avancé 🚀

Objectifs :

Le but de ce module est de vous faire réaliser un site web sous le CMS WordPress, l'intégralité des contenus sera administrable depuis le back-office de WordPress et réalisable par une personne lambda, j'entends par là un profil non technique qui ne lit et n'écrit pas de code. 😉

C'est aussi l'occasion de découvrir certains outils professionnels que nous utilisons en agence ainsi que d'autres technologies.

Important :

Le projet que nous allons réaliser est basé sur un projet professionnel existant, il est réalisé à des fins d'étude et apprentissage uniquement, je vous remercie d'avance de ne pas partager ou publier ce projet sur la toile.

Découverte des outils :

Je vous propose de découvrir les maquettes du projet à réaliser à l'aide de Zeplin (invitation envoyée par mail).

Zeplin permet d'effectuer la passation entre le pôle design qui réalise les maquettes et l'équipe de développement, les avantages sont multiples : gain de temps, récupération des styles, téléchargement des images utilisées sur la maquette…

Vous pouvez aussi définir un guide de styles, stocker l'ensemble des couleurs utilisées dans des variables qui vous serviront ensuite à mieux organiser vos styles CSS.

Si vous êtes familiarisé avec des outils qui proposent d'utiliser une grille pour réaliser votre mise en page comme Bootstrap ou Bulma par exemple, vous pouvez afficher la grille directement dans Zeplin.

Après la découverte et prise en main de Zeplin, je vous propose de passer à l'initialisation du projet. 😀

Initialisation du projet :

Nous allons sauvegarder votre travail à l'aide d'un logiciel de gestion de versions : Git

1) Initialiser un projet GitHub privé vide et clonez le ensuite sur votre machine
2) Télécharger et extraire les sources de WordPress dans votre projet : https://fr.wordpress.org/download/
3) Éditer le fichier .gitignore de votre projet pour y ajouter le fichier wp-config.php
4) Lancer l'installation de WordPress
5) Réaliser votre premier commit :

```
git add -A
git commit -m "init project"
git push
```

6) Connectez-vous à l'interface d'administration de WordPress : https://exemple.com/**wp-admin**

Thème starter :

Par gain de temps, nous allons travailler à partir d'un thème starter, c'est un thème qui inclut des fonctionnalités pré-configurées comme la gestion des menus, architecture, scss, template twig, transpilation du javascript…etc

Télécharger et installer le thème depuis l'adresse https://github.com/aslv11/works/blob/master/wordpress/themes/mmi.zip

Le thème utilise des dépendances, pour les installer, depuis le terminal allez dans le dossier du thème et lancez les commandes :

```
composer install
```

```
npm install
```

L'installation de toutes les dépendances peut prendre un moment, profitez en pour découvrir l'architecture du thème, nous retrouvons les templates standards WordPress telles que définies dans la documentation : https://developer.wordpress.org/themes/basics/template-hierarchy/

La différence est que notre thème, utilise le moteur de template Twig et nous offre ainsi la possibilité de ne plus mélanger le PHP et l'HTML.
Si vous ouvrez une template PHP, vous verrez qu'elles utilisent une méthode pour afficher le contenu à l'aide d'une template Twig.

```
Timber::render('Templates/index.twig', $context)
```

Le second paramètre appelé ici context est généralement un tableau qui contient les données à afficher dans la template twig.

Tips Twig :

Afficher une variable ou l'équivalent du echo

```
{{ title }}
```

Exécuter une instruction (if, else, for, include, extends…)

```
{% if 'you' != 'student' %}
  <p>Hello !</p>
{% endif %}
```

```
{% include 'Partials/Components/_pagination.twig' %}
```

```
{% for post in posts %}
    {{ post.title }}
{% endfor %}
```

Documentation : https://twig.symfony.com/

Styles SCSS :

Le SCSS est une version évoluée du CSS et est un langage compilé qui génère du CSS natif, les avantages sont multiples : variables, mixins, fonctions, imbrication du code…

Notre starter thème embarque le SCSS, vous trouverez les sources (les fichiers dans lesquels vous travaillerez et qui serviront uniquement à générer la feuille de styles définitive) dans le dossier assets/css/src.

Nous avons différents dossiers que je vous invite à découvrir, le dossier settings par exemple, contient la feuille de style _colors.scss, où vous pourrez stocker toutes les couleurs dont vous avez besoin (Rappelez-vous de Zeplin 😉) puis utiliser directement le nom de la variable dans vos styles, exemple :

```
.my-title{ color: $blue; }
```

Le fichier main.scss centralise tous les fichiers à importer, quand vous créerez un nouveau fichier, par exemple components/_nav-main.scss pour mettre en place les styles de votre menu principal, il faudra penser à l'importer dans main.scss

L'idée est d'avoir une approche composant, c'est à dire que quand vous créez un nouvel élément (le menu principal composant), on crée un nom de classe qui lui est propre ainsi qu'une feuille de style associée (_nav-main.scss), nous aurons ainsi un fichier dédié uniquement aux

styles du menu principal, cela est plus facile à relire et à maintenir quand nous avons beaucoup de composants dans un projet.

Aussi, je vous invite à choisir des noms pertinents quand vous nommerez vos composants de manière à savoir ce qu'ils sont juste en lisant leurs noms.

Voir : http://getbem.com/introduction/
https://sass-lang.com/

Note : Comme il s'agit d'un langage compilé, nous avons besoin d'une tâche qui se charge d'observer nos fichiers de travail et compiler lorsque des changements sont détectés, cela tombe bien, nous avons une tâche prévue pour, celle-ci est à lancer lorsque vous travaillez, elle tourne ainsi en tâche de fond, ci-dessous la commande à lancer

```
gulp
```

Le JavaScript :

Les sources du JavaScript se trouvent dans le dossier assets/js/src
En exécutant gulp, il va aussi observer les fichiers JS et transpiler le tout pour générer un seul fichier : main.js

Pourquoi ? Cela nous permet d'écrire du code JavaScript de nouvelle génération dit ES6 qui sera transpilé en ES5 pour qu'il soit compatible sur les navigateurs qui n'interprète pas le code ES6.

Voir : https://www.w3schools.com/js/js_es6.asp

Note : si vous le souhaitez, vous pouvez utiliser jQuery ou n'importe quelle librairie de votre choix (React ? 😉)

Amusez-vous !

Avant de commencer à réaliser du code, je rappel les principes fondamentaux du travail en équipe, oui vous êtes une équipe !

Soyez bienveillant et entraidez-vous, au cas où vous auriez de la difficulté à vous concentrez, je vous invite à découvrir la méthode Pomodoro : https://fr.wikipedia.org/wiki/Technique_Pomodoro

Je vous invite (vraiment 😀) à prendre le temps à l'aide d'une feuille, du tableau ou l'outil de votre choix pour réfléchir à la façon dont vous allez réaliser l'ensemble du projet.

Consulter les documentations quand cela vous semble nécessaire, être développeur c'est aussi rechercher et tester. N'hésitez pas à me demander si vous souhaitez que l'on revoit ensemble certains principes (JS, PHP, HTML, CSS…).

Bon travail ! 😀