

Tailwind

Introduction

Dans le cadre de ce projet, l'objectif était d'utiliser Tailwind CSS pour développer une interface web responsive en gérant les styles directement dans le HTML, sans avoir recours aux fichiers CSS classiques. Le sujet demandait d'explorer la configuration et l'utilisation de Tailwind à travers un mini projet Laravel, ainsi que d'implémenter différentes fonctionnalités comme la création d'une grille et d'une barre de navigation.

Objectifs et demandes

Les demandes spécifiques du sujet incluait :

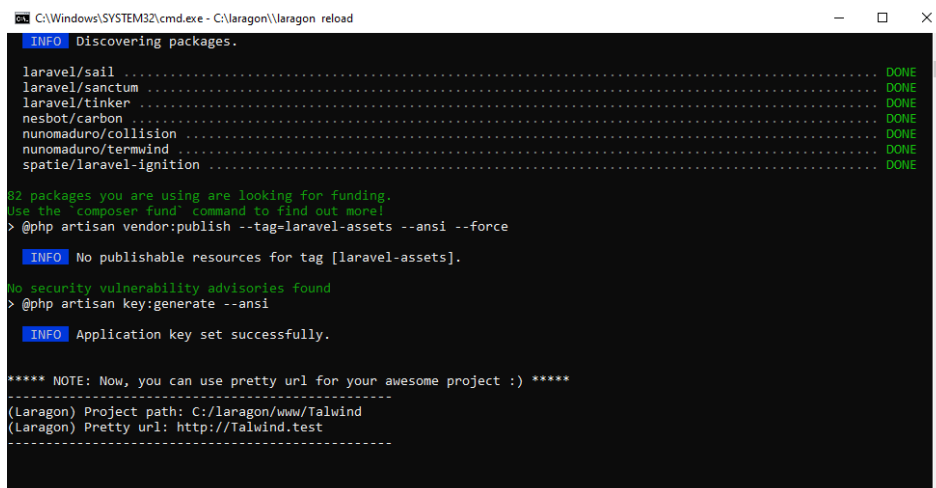
1. Installation et configuration de Tailwind CSS.
2. Création d'une grille responsive pour la page d'accueil.
3. Implémentation d'une barre de navigation utilisant Flexbox.
4. Réponse à des questions sur les classes utilisées et leurs fonctions dans le code.

Travail effectué

1. Installation de Tailwind CSS

J'ai d'abord installé Tailwind CSS en suivant les étapes :

- Installation via npm des dépendances tailwindcss, postcss, et autoprefixer.



```
C:\Windows\SYSTEM32\cmd.exe - C:\laragon\laragon reload
[INFO] Discovering packages.
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE

82 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

[INFO] No publishable resources for tag [laravel-assets].

No security vulnerability advisories found
> @php artisan key:generate --ansi

[INFO] Application key set successfully.

***** NOTE: Now, you can use pretty url for your awesome project :) *****
(Laragon) Project path: C:\laragon\www\Tailwind
(Laragon) Pretty url: http://Tailwind.test
```

```

C:\laragon\www\Talwind>npm install -D tailwindcss postcss autoprefixer
added 139 packages, and audited 140 packages in 7s

36 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\laragon\www\Talwind>npx tailwindcss init -p

Created Tailwind CSS config file: tailwind.config.js
Created PostCSS config file: postcss.config.js

C:\laragon\www\Talwind>

```

- **Modification du fichier tailwind.config.js pour inclure les fichiers nécessaires (Blade, JavaScript, Vue).**

```

JS tailwind.config.js X
JS tailwind.config.js > ...
1  /** @type {import('tailwindcss').Config} */
2  export default {
3    content: [
4      "./resources /* */ *.blade.php ",
5      "./resources /* */ *.js ",
6      "./resources /* */ *.vue "
7    ],
8    theme: {
9      extend: {},
10   },
11   plugins: [],
12 }
13

```

- **Ajout des directives @tailwind base, @tailwind components, et @tailwind utilities dans le fichier CSS.**

```

# app.css 3 X
resources > css > # app.css
1  @tailwind base;      Unknown at rule @tailwind
2  @tailwind components;  Unknown at rule @tailwind
3  @tailwind utilities;   Unknown at rule @tailwind
4

```

Cela a permis de configurer Tailwind et de pouvoir l'utiliser dans les fichiers Blade de Laravel.

Ce que j'ai appris : J'ai appris à installer et configurer une bibliothèque CSS utilitaire dans un projet Laravel en utilisant des commandes npm et à modifier des fichiers de configuration pour intégrer correctement Tailwind CSS dans un projet web.

2. Création de la grille responsive

Ensuite, j'ai intégré une grille sur la page d'accueil en utilisant les classes Tailwind :

- Utilisation de `grid-cols-4` pour définir 4 colonnes.

```
welcome.blade.php
resources > views > welcome.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     @vite ('resources/css/app.css')
7   </head>
8   <body class="container mx-auto">
9     <div class="grid grid-cols-4">
10      <div class="text-center border-2 border-blue-600">Colonne 1</div>
11      <div class="text-center border-2 border-blue-600">Colonne 2</div>
12      <div class="text-center border-2 border-blue-600">Colonne 3</div>
13      <div class="text-center border-2 border-blue-600">Colonne 4</div>
14    </div>
15  </body>
16 </html>
17
```

Colonne 1	Colonne 2	Colonne 3	Colonne 4
-----------	-----------	-----------	-----------

- Ajout de `gap-2` pour gérer les espacements entre les colonnes.

Cela m'a permis de structurer la page avec le système de grille intégré de Tailwind. J'ai aussi rendu la grille responsive en utilisant les breakpoints pour que la mise en page s'adapte aux différentes tailles d'écran.

```
welcome.blade.php X
resources > views > welcome.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     @vite ('resources/css/app.css')
7   </head>
8   <body class="container mx-auto">
9     <div class="grid gap-2 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4">
10      <div class="text-center border-2 border-blue-600">Colonne 1</div>
11      <div class="text-center border-2 border-blue-600">Colonne 2</div>
12      <div class="text-center border-2 border-blue-600">Colonne 3</div>
13      <div class="text-center border-2 border-blue-600">Colonne 4</div>
14    </div>
15  </body>
16 </html>
17
```

Colonne 1
Colonne 2
Colonne 3
Colonne 4

Ce que j'ai appris : J'ai appris à utiliser les classes Tailwind pour créer des mises en page structurées avec un système de grille flexible et à rendre ces mises en page adaptatives en fonction des différentes tailles d'écrans grâce aux breakpoints.

3. Barre de navigation responsive

Pour la navigation, j'ai utilisé le système Flexbox de Tailwind avec les classes :

- **flex** pour aligner les éléments horizontalement,
- **justify-between** et **items-center** pour ajuster l'espacement et l'alignement vertical.

```

welcome.blade.php X item.blade.php 2
resources > views > welcome.blade.php > ...
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      @vite ('resources/css/app.css')
7    </head>
8    <body class="container mx-auto">
9      <nav class="bg-red-500">
10       <div class="px-2 sm:px-4">
11         <div class="flex items-center justify-between h-14">
12           
13           <div class="hidden md:block space-x-2">
14             <x-item title="Accueil" active="true" />
15             <x-item title="Ventes" />
16             <x-item title="Clients" />
17             <x-item title="Produits" />
18           </div>
19         </div>
20       <div class="md:hidden">
21         <div class="px-2 pb-2 space-y-1">
22           <x-item title="Accueil" active="true" block="true" />
23           <x-item title="Ventes" block="true" />
24           <x-item title="Clients" block="true" />
25           <x-item title="Produits" block="true" />
26         </div>
27       </div>
28     </div>
29   </nav>
30
31 </body>
32 </html>

```

La classe `hover:bg-red-800` a été utilisée pour créer un effet de survol. J'ai appris à structurer une barre de navigation qui s'adapte aux tailles d'écran, change de disposition en mode mobile, et utilise des interactions simples comme le changement de couleur au survol.



Ce que j'ai appris : J'ai appris à utiliser Flexbox dans Tailwind pour aligner et espacer correctement les éléments d'une barre de navigation, à gérer la transition entre un menu horizontal et un menu vertical pour les petits écrans, et à ajouter des interactions visuelles comme le survol.

4. Explications sur les classes utilisées

Les classes de Tailwind présentes dans la page welcome servent à définir rapidement l'apparence et la mise en page. Par exemple, la classe `bg-gray-300` applique un fond gris, et `p-12` crée des marges internes. Cela permet de gérer les styles directement dans le HTML sans avoir à séparer la structure du style.

Ce que j'ai appris : J'ai appris à utiliser des classes utilitaires Tailwind pour contrôler directement l'apparence des éléments HTML sans avoir besoin de créer des fichiers CSS séparés, ce qui simplifie grandement le processus de stylisation.

Conclusion

Ce projet m'a permis d'acquérir une meilleure compréhension de l'utilisation de Tailwind CSS pour créer des pages web responsives. L'une des principales difficultés que j'ai rencontrées concernait l'utilisation de Visual Studio Code. Je n'avais pas la bonne version du logiciel, ce qui posait des problèmes de compatibilité. De plus, certaines extensions nécessaires pour travailler avec Tailwind CSS n'étaient pas installées. J'ai donc dû mettre à jour Visual Studio Code et installer les extensions requises, ce qui a retardé le début du projet.

Malgré cela, j'ai appris à utiliser efficacement Flexbox, les systèmes de grille et les breakpoints pour rendre la mise en page responsive. En plus de ce projet, j'avais déjà réalisé des projets avec Bootstrap, ce qui m'a permis de comparer les deux bibliothèques. Tailwind offre plus de flexibilité pour personnaliser les styles, tandis que Bootstrap est plus rapide à prendre en main grâce à ses composants prédéfinis.