**Réalisation d’une application « TODOList »**

1. **Le projet**

Je dois développer une application single page permettant d’éditer une liste de taches stockées dans une base de données relationnelles.

Je décompose cette application en 3 pages :

* La page principale liste l’ensemble des taches
* La deuxième page permet d’en afficher le détail
* La dernière page offre la possibilité de modifier

1. **Maquettage**

Dans un premier temps je crée la maquette, afin de construire une mise en page adaptée à mon objectif et contenu.

Travaillant seul sur le projet, je commence par dessiner sur feuilles libres les différentes parties qui constituent chacune des interfaces web.

Je commence par la vue mobile pour terminer par la vue desktop.

1. **Structure d’un document HTML**

Je démarre mon éditeur de code et ajoute ma première page, j’inclus la structure de base d’une feuille html le ***Doctype***, la balise ***html*** composé d’une section ***head*** et ***body.***

1. **Mise en forme des documents HTML**
   1. **CSS**

Pour mettre en forme mes pages web je peux utiliser le concept de feuille de style CSS (*Cascading Style Sheets)*,langage déclaratif.

Une feuille de style est structurée en règles CSS basées sur le même modèle.

Elle se décompose en un sélecteur indiquant au navigateur les éléments HTML qui doivent être mis en forme, appelées bloc de déclaration se trouvant entre accolades.

Ceux-ci renferment des propriétés sous forme clé valeurs, permettant de modifier l’aspect des éléments HTML.

* 1. **BOOTSTRAP Framework CSS**

J’ai choisi d’utiliser les classes de la bibliothèque Bootstrap car il propose une alternative au développement CSS pur.

Son principal avantage réside dans l’amélioration de la productivité de développement.

En effet, il corrige les problèmes de compatibilité entre navigateurs (cross-browser) offrant une présentation quasi identique, normalise la présentation en proposant un ensemble homogène de styles, propose un système de grille pour faciliter le positionnement des éléments, est responsive et adaptable autrement dit, les blocs de contenus (textes et images) se redimensionnent et se réorganisent en fonction de l’espace disponible sur l’écran.

Par ailleurs, il propose des éléments stylisés et des composants qui nécessitent l'utilisation de JavaScript pour fonctionner.

* 1. **L’installation du framework**

Tout d’abord, je dois installer sa bibliothèque, plusieurs méthodes sont disponibles :

* Télécharger une version des fichiers nécessaire au fonctionnement de Bootstrap (bootstrap.css, jquery.js, popper.js, bootstrap.js),

Pour le rendre fonctionnel je dois l’ajouter à mon projet, dans ce cas j’insère une balise ***link*** dans l’entête de ma page section ***head*** pointant vers le fichier CSS de Bootstrap, puis j’ajoute dans une balise script les 3 liens vers les fichiers JavaScript.

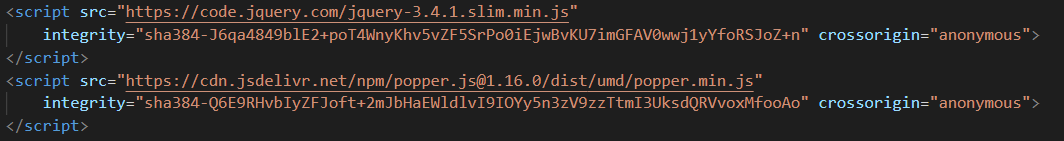
* Avec le CDN (content delivery network), qui est constitué d’ordinateurs reliés en réseau à travers Internet et qui coopèrent afin de mettre à disposition du contenu ou des données à des utilisateurs, j’utiliserai la même méthode déclarative citée précédemment.
* L’ajouter via le gestionnaire de paquets officiel de Node.js ***npm*** avec cette commande **npm install bootstrap**, je bénéficie en plus de la version portable en ***Sass***, *préprocesseur CSS* qui permet de personnaliser Bootstrap mais aussi d’utiliser des fonctionnalités comme des variables, l’imbrication de règles CSS, des fonctions, l’import de fichiers, les mixins… simplifiant le développement et la maintenabilité.

Cependant, le code nécessite une compilation des fichiers ***scss*** en ***css*** car les navigateurs n’intègrent pas ce langage.

C’est cette dernière solution que je choisi. Je crée un dossier SASS, j’ajoute un fichier **main.scss,** j’importe le fichier sass de bootstrap

@import "node\_modules/bootstrap/scss/bootstrap.scss";

suivi des règles personnalisées qui viennent en complément des fonctionnalités de Bootstrap, sans oublier d’ajouter dans une balise script les 2 liens vers les fichiers JavaScript.



Pour la compilation du fichier SASS je saisi dans le fichier **package.json** (fichier répertoriant les dépendances) le nom de la source et la destination du fichier à compiler.

  "scripts": {

    "compile-sass": "node-sass sass/main.scss assets/css/main.css --watch",

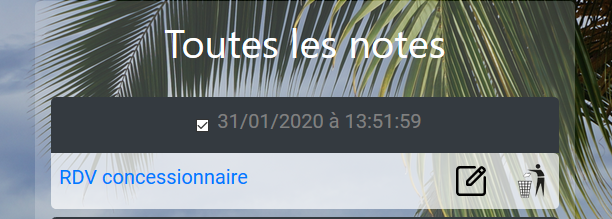
    "test": "echo \"Error: no test specified\" && exit 1"

  },

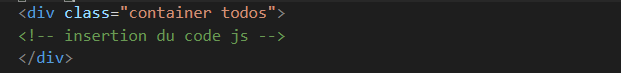
* 1. **Le principe des grilles**

Pour constituer mes pages, j’utilise le principe des grilles de Bootstrap, un élément conteneur (container) englobant des lignes (row) elles-mêmes subdivisées jusqu’à 12 colonnes (col).

Voici un exemple pour la réalisation d’une tache (une vue mobile)

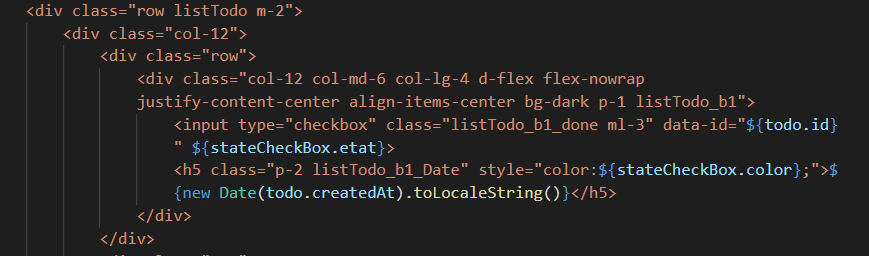
****

Pour créer ce design j’ai placé le container dans la page d’accueil, le code html est chargé dynamiquement en JavaScript.



Afin que ma page s’adapte au différents types de support numériques j’utilise les classes colonnes, 5 tailles sont proposées du mobile au écran ultra large.

* col- : Extra Small
* col-sm : Small
* col-md : Medium
* col-lg : Large
* col-xl : Extra Large



Ci-dessus dans cette portion de code on définit la taille d’une colonne « col-12 col-md-6 col-lg-4 », celle-ci s’étire sur toute la largeur de son conteneur sur un support mobile puis se réduit de moitié sur une tablette et enfin elle occupera 1/3 de la taille d’un écran large.

* 1. **Utilisation du composant modal**

Afin d’ajouter une nouvelle tâche je me suis servi d’un plug-in ou composant le modal, il permet d’afficher une boite de dialogue il utilise HTML, CSS, JavaScript.

Je copie un des exemples fournis sur le site et modifie son contenu pour l’adapter à mes besoins en ajoutant dans le corps du modal un formulaire.

Le bouton avec l’attribut **data-target** pointant sur identifiant du modal permet le déclenchant de l’ouverture de la fenêtre.

La fermeture s’effectue via un bouton contenant l’attribut **data-dissmiss**.

