**Cahier des charges**

**d'une Application "Mobile-First"**

*But de cet exercice :* ***Faire travailler l'existant + API***

*Temps:* ***A faire sur 5 jours (Mercredi - Lundi)***

*Stacks*: MaterializeCss + Jquery + SASS + Gulp + Plugins Jquery + API + ES6/Module ES5

**Magasins de Sources D'aides**

* <http://jsfiddle.net/Braulio/vDr36/>
* <http://codepen.io/olivertaylor/pen/NNZjeo>
* <http://codepen.io/SitePoint/pen/vKXZrz>
* <https://www.sitepoint.com/load-flickr-photos-using-jsonapi/>
* <https://www.sitepoint.com/jsonp-examples/>
* <https://api.flickr.com/services/feeds/photos_public.gne>
* <http://materializecss.com/grid.html>
* <http://materializecss.com/media-css.html>
* <http://codepen.io/mike-north/pen/MwVoYp>
* <https://scotch.io/tutorials/make-material-design-websites-with-the-materialize-css-framework>
* <http://materializecss.com/side-nav.html>
* <http://materializecss.com/navbar.html>
* <http://tympanus.net/Development/GridLoadingEffects/>
* <http://tympanus.net/codrops/2013/07/02/loading-effects-for-grid-items-with-css-animations/>
* <https://www.flickr.com/services/api/>
* <http://www.alsacreations.com/article/lire/1402-web-storage-localstorage-sessionstorage.html>
* <https://www.npmjs.com/package/gulp-uncss>
* <http://thesassway.com/intermediate/using-source-maps-with-sass>
* <http://materializecss.com/preloader.html>
* <http://materializecss.com/transitions.html>

Etape 0/ Créer un Environnement à l’application

Créer un environnement avec Gulp et Bower "*Ready To Go*" avec:

+ L’environnement **GULP et ses plugins** en implémentant les plugins **gulp-size** (taille du fichier), puis **gulp-uncss** et mettre en place les **SourceMaps** a travers **gulp-sass**: <https://github.com/dlmanning/gulp-sass>

*Petite pause: Lire l'article suivant sur les SourceMap en Naviguateur:* [*https://developer.mozilla.org/fr/docs/Outils/D%C3%A9bogueur/Comment/Utiliser\_une\_source\_map*](https://developer.mozilla.org/fr/docs/Outils/D%C3%A9bogueur/Comment/Utiliser_une_source_map)

+ Créer **une architecture SASS** pour préparer la mise en forme de l’application à travers ses composants **Grille**, des **Cards**, la **SideNav** et le **Formulaire et les Inputs**

+ Créer un **Repository sur Github** “**AppFlickr**” et synchroniser votre environement avec GIT *(git init, git add origin, git add, git commit, git push...)* avec un README.MD a la racine en guise de notice. Faire un premier push de l’environnement :)

---------------------------------------------------------------------------------------------------------------------------

Etape 1/ Ajax avec l'API Flickr

1/ Créer un Menu Navbar de 5 Buttons centrés sur la pages: *Javascript - PHP - AngularJs - HTML - CSS*

2/ Quand je clique sur un boutton, cela m’ajoute une classe active sur le bouton et ça me charge des images en AJAX les photos Flickr avec la fonction $.ajax() dans une Grid de photos

4/ Mettre en, place une SideNav et a l’intérieur un formulaire avec 2 éléments en Materializecss. Le formulaire aura les champs suivants:

+ un champs input "Tags"

+ un bouton de soumission "Ajouter d'autre photos"

Quand je saisis un texte, cela m’ajoute un bouton “actif” au Menu et me charge les images en conséquant

5/ Afficher les tags saisies du formulaire avec des Chips de Materializecss <http://materializecss.com/chips.html>

6/ Mettre en mémoire les Tags saisies grâce a Session Storage

7/ Quand on arrive sur l'application, par défaut les tags se recharge ainsi que les photos associés

8/ Ajouter un Range Slider Materialize CSS qui me permet d'afficher un nombre limité d'items dans la Grid ( de 10 à 100)

**Magasin à Bonus**

*Bonus: Préparer en SASS les medias queries pour tablette, phablette et smartphone de l’application*

*Bonus: Ajouter “Pre-loader” une animation qui se lance au chargement des photos en AJAX*

*Bonus: Je peux ajouter plusieurs tags séparés par des virgules*

*Bonus: Ajouter le plugin Masonry et un effet au scroll http://tympanus.net/codrops/2013/07/02/loading-effects-for-grid-items-with-css-animations/*

*Bonus: Créer un classe Flickr en ES6 transpilé par Babel avec le module npm Babelify afind e factoriser les fonctionalités dans une classe qui gères tout le fonctionement logique de l’applications*