# DOSSIER DE SYNTHESE

Développeur Web et Web Mobile

**Sommaire**

[DOSSIER DE SYNTHESE 1](#_Toc52979052)

[I. Présentation du projet 3](#_Toc52979053)

[II. Etude concurrentielle 4](#_Toc52979054)

[1) Étude concurrentielle 5](#_Toc52979055)

[2) Conclusion Étude concurrentielle 5](#_Toc52979056)

[III. Recherche et développement 6](#_Toc52979057)

[1) Une page d’accueil claire, simple et précise : 6](#_Toc52979058)

[2) Des textes lisibles : 6](#_Toc52979059)

[3) Un affichage rapide : 7](#_Toc52979060)

[4) Un plan de page lisible : 7](#_Toc52979061)

[5) Des menus accessibles et intuitifs : 7](#_Toc52979062)

[6) Une hiérarchie visuelle des éléments : 7](#_Toc52979063)

[7) Un accès en 3 clics : 8](#_Toc52979064)

[8) Un accès optimisé sur les différents supports : 8](#_Toc52979065)

[IV. Spécifications techniques 8](#_Toc52979066)

[1) Liste des technologies utilisées : 8](#_Toc52979067)

[2) Tools de design : 11](#_Toc52979068)

[3) Environnement de travail : 12](#_Toc52979069)

[4) Outil de stockage : 12](#_Toc52979070)

[V. Charte graphique avec Logo 13](#_Toc52979071)

[VI. Maquette Adobe xD 14](#_Toc52979072)

[VII. Persona 14](#_Toc52979073)

[1) Persona Primaire 14](#_Toc52979074)

[2) Ux utilisateur primaire voir annexe 1 15](#_Toc52979075)

[3) Persona Secondaire 15](#_Toc52979076)

[4) Ux utilisateur secondaire 15](#_Toc52979077)

[VIII. Zoning des fonctionnalités 15](#_Toc52979078)

[IX. Présentation base de données 16](#_Toc52979079)

[1) Dictionnaire de données : 16](#_Toc52979080)

[X. Prototypage Bureau et Mobile 17](#_Toc52979081)

[XI. Prototypage de fonctionnalités pour la page Connexion 19](#_Toc52979082)

[1) Navigation après connexion et menu. 19](#_Toc52979083)

[XII. Description des principales fonctionnalités 20](#_Toc52979084)

[XIII. Contraintes techniques 21](#_Toc52979085)

[XIV. Sécurisation 22](#_Toc52979086)

[1) Hébergeur : 22](#_Toc52979087)

[2) Sécurisation des droits : 23](#_Toc52979088)

[XV. CSS 23](#_Toc52979089)

[XVI. JavaScript 24](#_Toc52979090)

[XVII. Projection du site 25](#_Toc52979091)

[XVIII. Conclusion 26](#_Toc52979092)

[XIX. Remerciements 27](#_Toc52979093)

[XX. Annexe 28](#_Toc52979094)

# Présentation du projet

Mon projet de fin d’étude est une futur application coté client, développé en java script avec la librairie React et le framework Meteor en utilisent node.js et mongoDB pour le modèle de donnée. Cette plateforme permettrait à l’utilisateur de pouvoir publier des vidéos sur le thème des jeux vidéo vers des plateformes spécifique de type Twitch, Youtube, Tweeter simultanément. Et un espace personnel développé autour de l’utilisateur. Pour le début du développement, nous utiliseront la plus simple des plateformes et la plus représentatives. Puis avec le temps nous agrémenterons de plateforme et de fonctionnalité.

La fonctionnalité principale est de pouvoir poster une vidéo courte sur plusieurs réseaux aux types et format adapté à celui-ci. Le facteur principal est la persistance de donnée sans hébergement mais avec un hébergement indirect.

Ce projet est un tremplin vers une succession d’activité qui nous permettrons avec les accords des différents distributeurs comme exemple facebook, instagram, snapchat. Pour crée des Clips de publication avec des outils d’éditions.

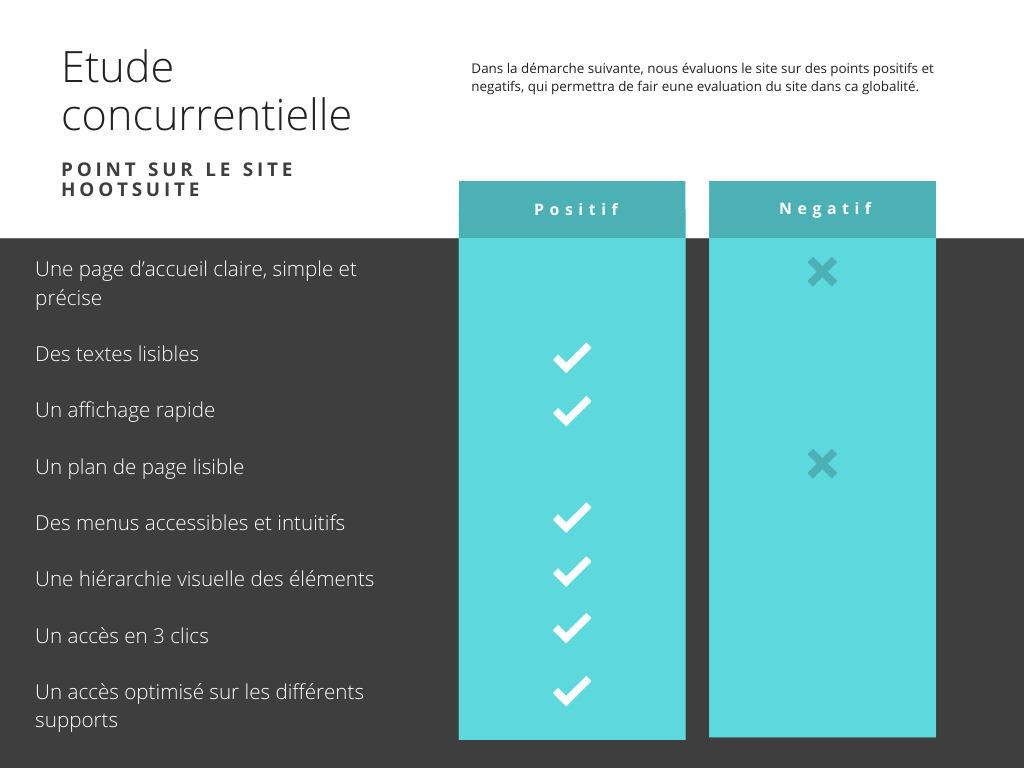
Avec l’ambition de crée un outil de management de medias pour le monde professionnel et public du jeu vidéo ou des activités médiatiques qui permettre aux différents acteurs de pouvoir organisés sont plannings de publications mais aussi de les différée.

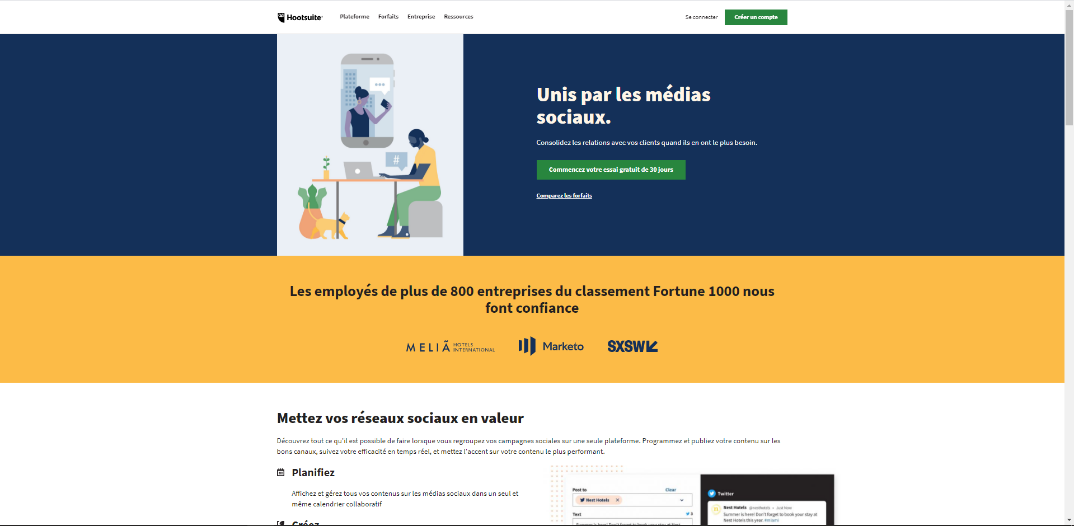
# Etude concurrentielle

## Étude concurrentielle

Site web <https://hootsuite.com/fr/>

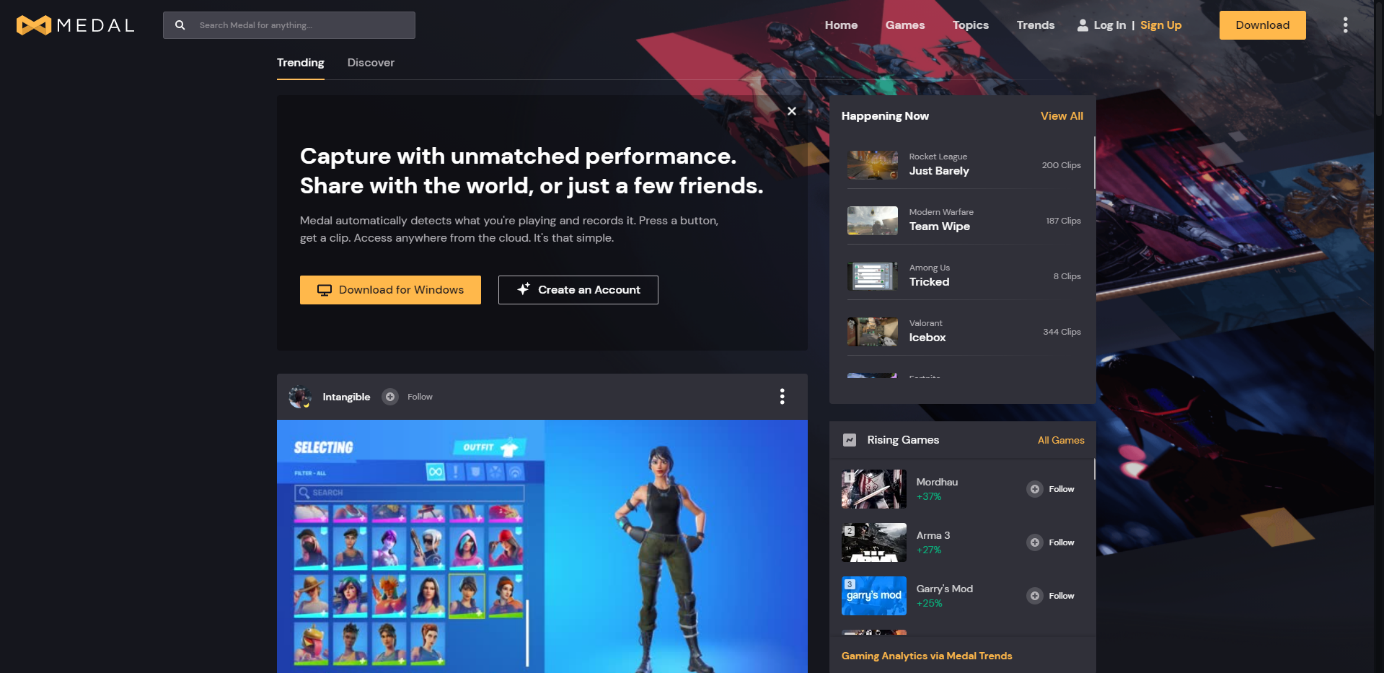
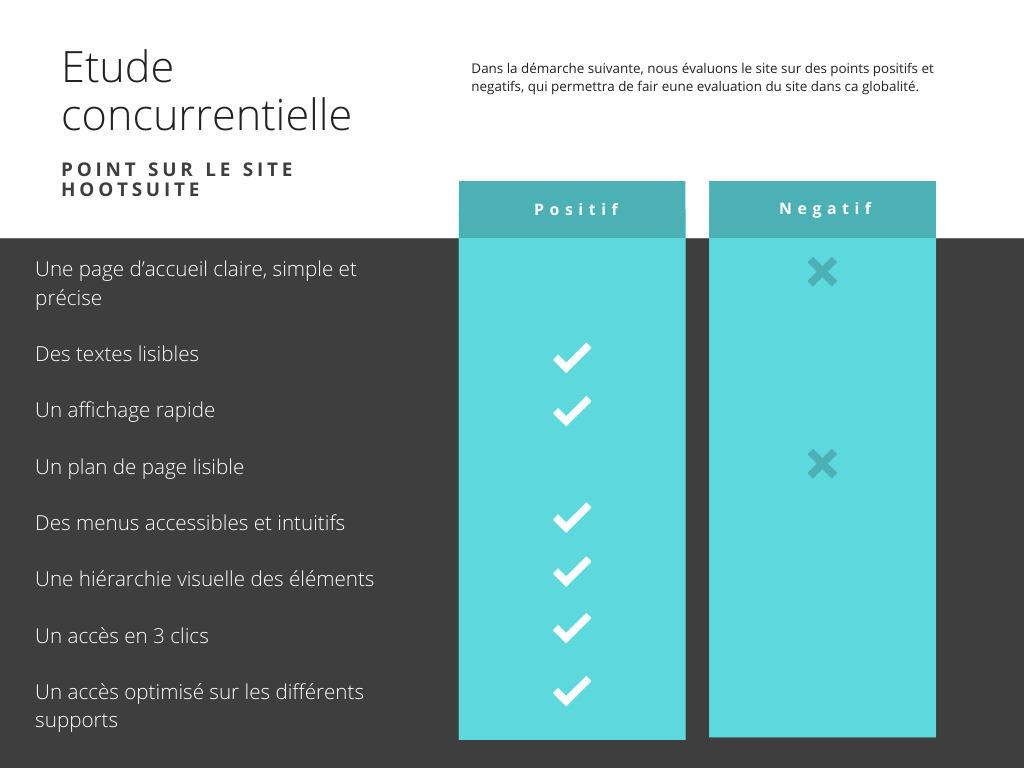
Ce site permet de prévoir les postes et l’organisation de partage de médias sur les réseaux sociaux. Son utilisation principale reste la gestion de partage de vidéo et de management de son planning de partage. Il utilise le principe fondamental que je veux mettre en valeurs, le partage de médias simultanément sur plusieurs plateformes mais dans la plus grande simplicité.



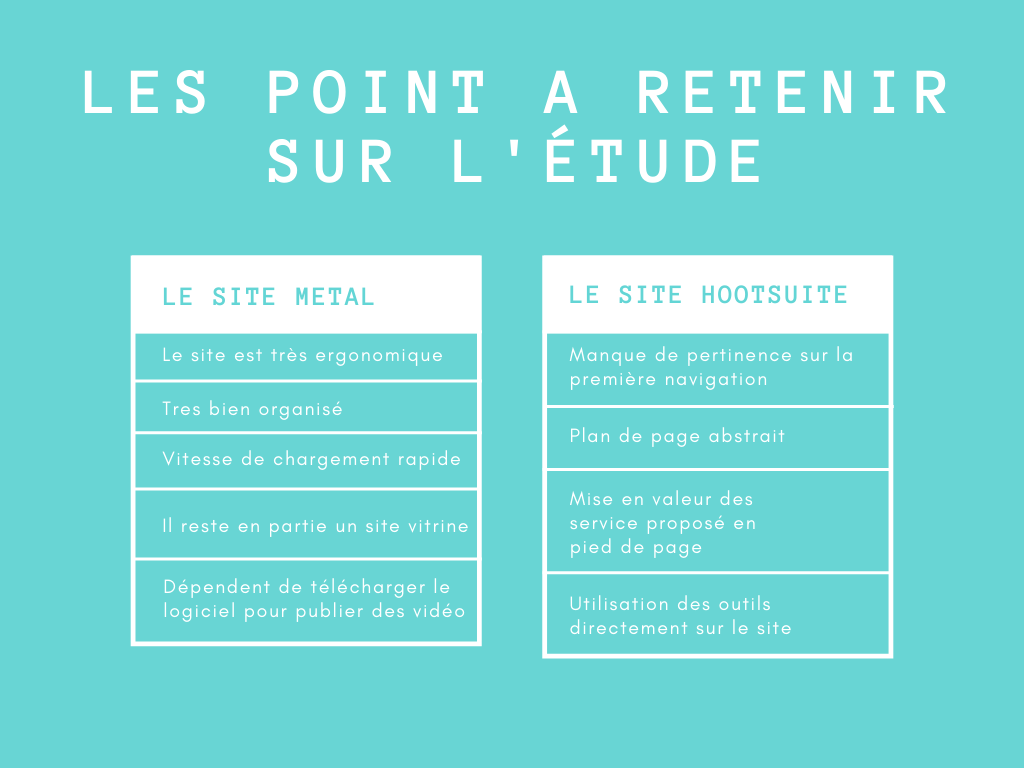


<https://medal.tv/>

Medal est le meilleur moyen d'enregistrer, de regarder et de partager des clips de jeu avec le plus faible impact FPS possible. Synchronisez les clips sur le cloud pour libérer de l'espace et créer votre profil. Avec un logiciel d’installation, le site permet de visionner les clips.



## Conclusion Étude concurrentielle



Les deux sites traites des sujets bien précis et ce complète. J’ai souligné plusieurs points intéressants sur l’organisation du site METAL.TV en gardent les principes de Hootsuite qui est de tout avoir disponible directement sur l’application Web. Je tien aussi a souligné la simplicité de navigation des deux sites. Je garde comme ligne de conduite la simplicité dans les domaines fonctionnel et visuelle.

# Règles adopté de l’Ux et Ui

## Une page d’accueil claire, simple et précise :

C’est la page qui aura bien souvent le plus fort trafic. Il est donc essentiel que les visiteurs aient tout de suite envie d’en savoir plus.

Contrairement à ce que l’on pourrait penser, ce n’est pas la beauté du design, qui poussera les visiteurs à l’action. Ils chercheront à accomplir une action ou trouver une information, dans la simplicité.

Le premier objectif est de leur simplifier la vie : faire simple. Bien sûr, l'ensemble des pages devront également respecter cette règle.

## Des textes lisibles :

Lire un texte sur un ordinateur fatigue rapidement les yeux alors il convient d’éviter les colonnes de texte trop larges ou trop fines, les polices illisibles ou trop exotiques, les textes sur fond gris ... Engagé sur un choix personnel, j’ai pris la décision d’utilisé une typographie spécialisé pour les dyslexique ou les personnes avec des troubles de la lecture. Etant moi-même dyslexique, je souhaite apporté des facilitées et intégré des outils de lecture-t-elle que celui-ci.

Voici les points a privilégiez:

* Texte foncé sur fond claire ou inverse
* Une taille de police standard (pas de police microscopique ou macroscopique).
* Une police spécifique
* le soulignage des liens exclusivement
* des plans de page découpés en paragraphes avec titres, sous titres

## Un affichage rapide :

Les internautes sont des gens pressés en règle générale. Je ne souhaite pas que l’internaute ressorte aussi vite qu’il est entré, j’optimiserais le poids des pages.

## Un plan de page lisible :

Chaque page doit avoir un titre et être aérée par une mise en page agréable. Avec une utilisation des sous-titres, des paragraphes courts et concis.

## Des menus accessibles et intuitifs :

Les visiteurs ne devront pas passer plus 10 secondes à tenter de comprendre comment fonctionne le site. Les menus doivent être clairs et accessible au premier coup d’œil.

## Une hiérarchie visuelle des éléments :

Afin de capter l’œil de nos visiteurs, il faudra jouer sur les contrastes des zones les plus importantes du site. Mais attention de ne pas en abuser au risque de leur donner le tournis. Le teste à suivre : flouter légèrement une page avec Photoshop et constater ou se porte le regard en priorité.

## Un accès en 3 clics :

L’utilisateur pourra en trois clics avoir accès à tout le contenu du site depuis la page principale.

## Un accès optimisé sur les différents supports :

A l'heure où plus de la moitié des navigations s’effectuent sur un appareil mobile, votre site doit offrir une expérience optimale sur chacun de ces supports. Pour se faire, le responsive design permettra de faire varier automatiquement le mode d’affichage des pages en fonction de la taille de l’écran.

# Recherche et développement

# **Les outils utilisés pour la recherche et le développement, ont été**.

* Une maquette Adobe xD
* Personna (<https://uxpressia.com/> )
* Ux Maps (<https://uxpressia.com/> )

## Liste des technologies utilisées :

* + 1. Les logiciels de développement :
* Microsoft Visual Studio code est une suite de logiciels de développement pour Windows et mac OS conçue par Microsoft. Elle facilite le développement d'applications pour toutes les plates-formes et tous les langages. La dernière version s'appelle Visual Studio code 2019. <https://visualstudio.microsoft.com/fr> .
  + 1. Les langages utilisés :
* HTML5 Dernière révision majeure du HTML. Cette version a été finalisée le 28 octobre 2014.
* CCS3 Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.
* JavaScript Un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation de Node.js
  + 1. Framework et librairie :
* React.js souvent appelé React ou ReactJS est une bibliothèque JavaScript responsable de la construction d’une hiérarchie de composants de l’interface utilisateur ou, en d’autres termes, responsable de la génération des composants de l’interface utilisateur. Il fournit un support pour front-end et côté serveur. Rappelez-vous que React.js est le V du pattern MVC, c’est donc une bibliothèque uniquement destinée à générer vos vues.
* Meteor permet de développer avec le même [langage](https://fr.wikipedia.org/wiki/Langage_de_programmation) (en [Javascript](https://fr.wikipedia.org/wiki/Javascript) ou dans un langage compilant vers Javascript comme [CoffeeScript](https://fr.wikipedia.org/wiki/CoffeeScript" \o "CoffeeScript) ou [Dart](https://fr.wikipedia.org/wiki/Dart_(langage_informatique))) et avec la même [API](https://fr.wikipedia.org/wiki/Interface_de_programmation) sur le client et sur le serveur. Ce choix d’architecture permet de déplacer facilement un traitement du serveur vers le client (et réciproquement) voire de le dupliquer par exemple dans le cas de la validation d'un formulaire.

Dans cette logique, Meteor inclut un [système de gestion de base de données](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_base_de_donn%C3%A9es) côté client, fonctionnalité originale du framework. Il est ainsi possible d'effectuer des requêtes même en étant déconnecté du serveur. Cela permet notamment à Meteor d'inclure par défaut, des mécanismes de *compensation de latence*. Par exemple, l'envoi d'un message dans un chat sera instantanément ajouté au fil des messages au clic sur le bouton "Envoyer", tandis que la vérification du message se fera en arrière-plan côté serveur. Ce mécanisme permet l'utilisation de la [programmation réactive](https://fr.wikipedia.org/wiki/Programmation_r%C3%A9active) côté client.

* Node.js Meteor s'appuie sur Node.js qui permet d’exécuter du code JavaScript sur le serveur. Cela signifie que nous allons utiliser le même langage de programmation sur le client et sur le serveur, un confort indéniable pour le programmeur. Meteor pousse même l'idée un peu plus loin en unifiant les API utilisées sur le client et sur le serveur, on parle de JavaScript isomorphique, un terme un peu pédant pour désigner ce concept qui vous semblera bientôt naturel. Prenez par exemple la méthode Meteor.startup qui permet d’exécuter du code au démarrage du client ou du serveur. Du côté serveur la fonction est exécutée dès que le processus Node.js est lancé, et du côté client elle l'est dès que le DOM est prêt à être modifié. Il s'agit donc d'une méthode disponible sur les deux environnements pour le même usage, exécuter du code au démarrage, mais implémentée de manière différente sur le client et sur le serveur.
* Mongo DB C'est dans cet esprit que Meteor propose également une base de données complète accessible du côté client. À l'usage vous verrez qu'il est très commode de pouvoir faire des requêtes directement dans le navigateur avec la même API que sur le serveur. Il s'agit de l'API de MongoDB qui est la base de données utilisée par défaut avec Meteor. Un développeur utilisera donc les mêmes méthodes sur deux environnements différents pour exprimer la même idée (« stocker des données ») bien que l'implémentation de ces méthodes diffère largement en fonction des contraintes imposées à l'environnement.

**AVANTAGES DE REACT , METEOR , NODE.JS**

DOM (document object model) est un compromis des vus sur les entrées et sorties de données. Le DOM virtuel de React est plus rapide que le modèle de rafraîchissement complet conventionnel, puisque le DOM virtuel ne rafraîchit que certaines parties de la page. Ce qui est intéressant, c’est que l’équipe de Facebook n’était pas consciente qu’une actualisation partielle d’une page se révélerait plus rapide. Facebook cherchait juste un moyen de réduire leur temps de reconstruction, et le rafraîchissement partiel du DOM était juste une bonne solution. Au final, cela augmente les performances et accélère la programmation. Vous pouvez réutiliser des composants de code dans React JS, ce qui vous fait gagner beaucoup de temps. La génération complète de vos pages, du serveur au navigateur, améliorera le référencement de votre application web.

Il améliore la vitesse de débogage, facilitant ainsi la vie de votre développeur.

Même pour ceux qui ne sont pas familiers avec React, il est facilement lisible. De nombreux frameworks exigent que vous appreniez une longue liste de concepts qui ne sont utiles que dans le framework. React s’efforce de faire le contraire.

Vous profitez de toutes les avancées du langage Java et de son écosystème.

Environnement de développement :

## Tools de design :

* *PhotoShop* est un logiciel de retouche, de traitement et de dessin assisté par ordinateur. Édité par Adobe, il est principalement utilisé pour le traitement des photographies numériques, mais sert également à la création. J'ai pu donc faire murir les idées de John melis avec cet outil et pouvoir commencer à intégrer la charte graphique.
* *Adobe xD* est un outil vectoriel développé et publié par Adobe Inc. pour la conception et le prototypage de l'expérience utilisateur pour les applications Web et mobiles. Il m'a permis de faire un prototypage proche de la réalité et de tester la navigation.
* *Draw.io pro est une application de création de diagrammes compatible avec Google,* que j’ai utilisé afin de mettre en place le zonning des fonctionnalisées.

## Environnement de travail :

* Ordinateur disponible à l'école Ariès
* Ordinateur disponible chez moi avec

➥ **MSI Z170A KRAIT GAMING 3X**  
➥ **Intel Core i5-6600K (3.5 GHz)**  
➥ **MSI GeForce GTX 980 Ti GAMING 6 Go**  
➥ **DDR4 HyperX Savage Black, 2 x 8 Go, 3000 MHz**  
➥ **SSD Samsung Série 850 EVO, 250 Go**  
➥ **BenQ 24" LED - XL2411Z 144Hz**   
➥ **Corsair RGB k70 lux**  
➥ **BenQ ZOWIE FK1+ ➭ Dpi 400  
➥ HeadPhone HyperX cloud I**

* Responsivité avec le simulateur navigateur Google mode Smartphone 5S.
* Mon téléphone personnel S8+.
* Mise en ligne avec Reseau Orange fibre (800mb/s débit descendent 300mb/s débit montant).

## Outil de stockage :

* Base de donnée MongoDB dont les appels sont fait et gérée par meteor.

# Charte graphique

# Polices / Fonts

La police intégrée dans le projet est de Google font

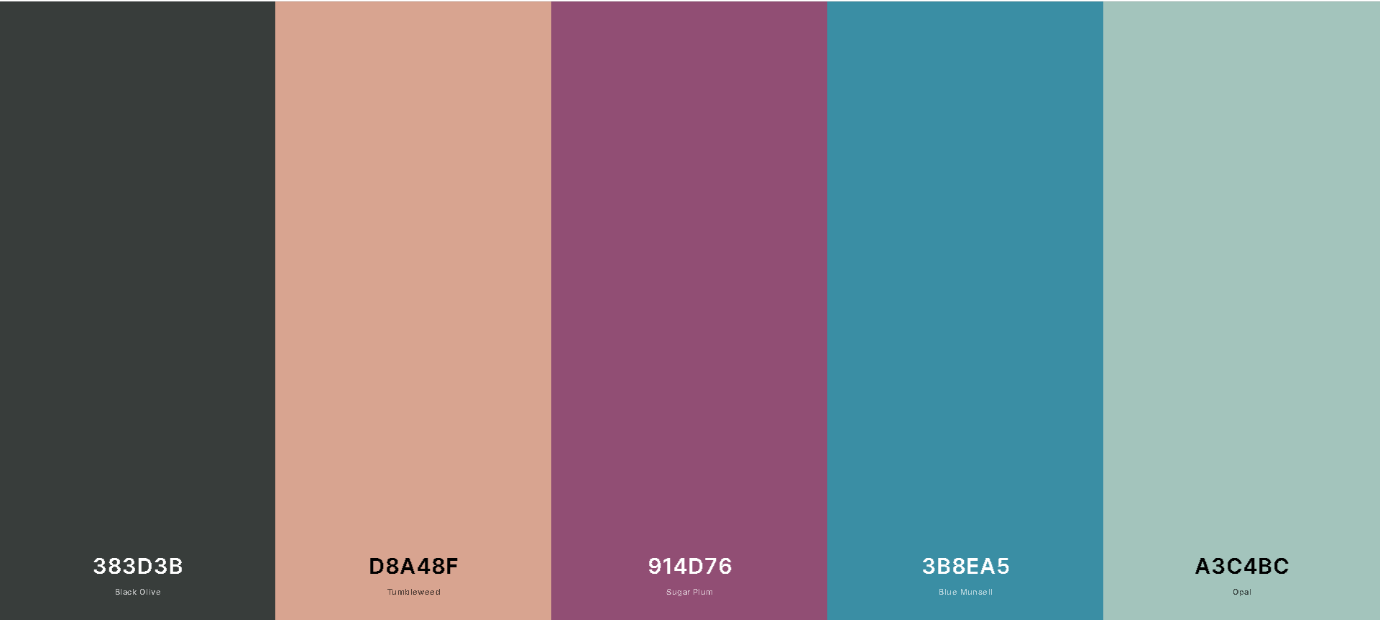
# Tailles natives

Les tailles natives du projet sont sur des chiffres ronds ou des multiples de 8px

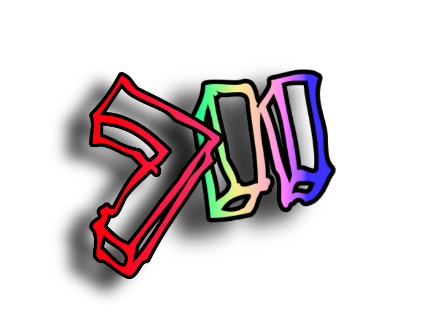
Dans le cas des références en REM elles seront présentées de 0.5 en 0.5

# Couleurs

Des variables on était mise en place pour faciliter utilisations.



# Logos



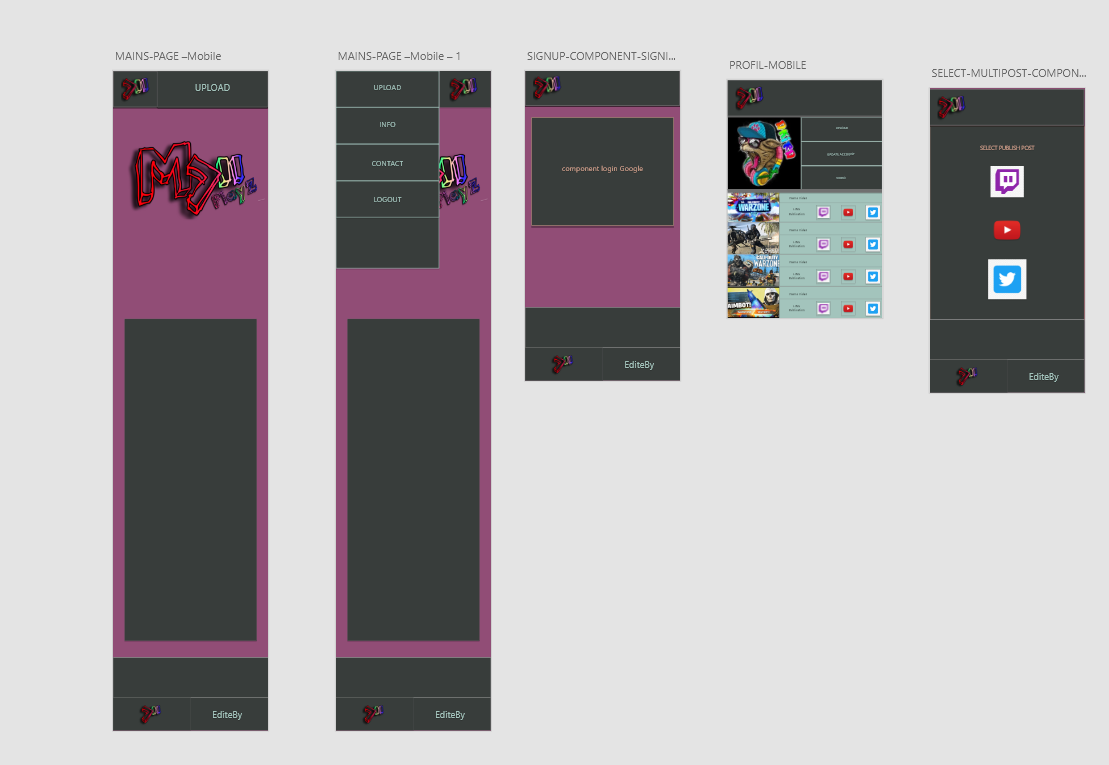




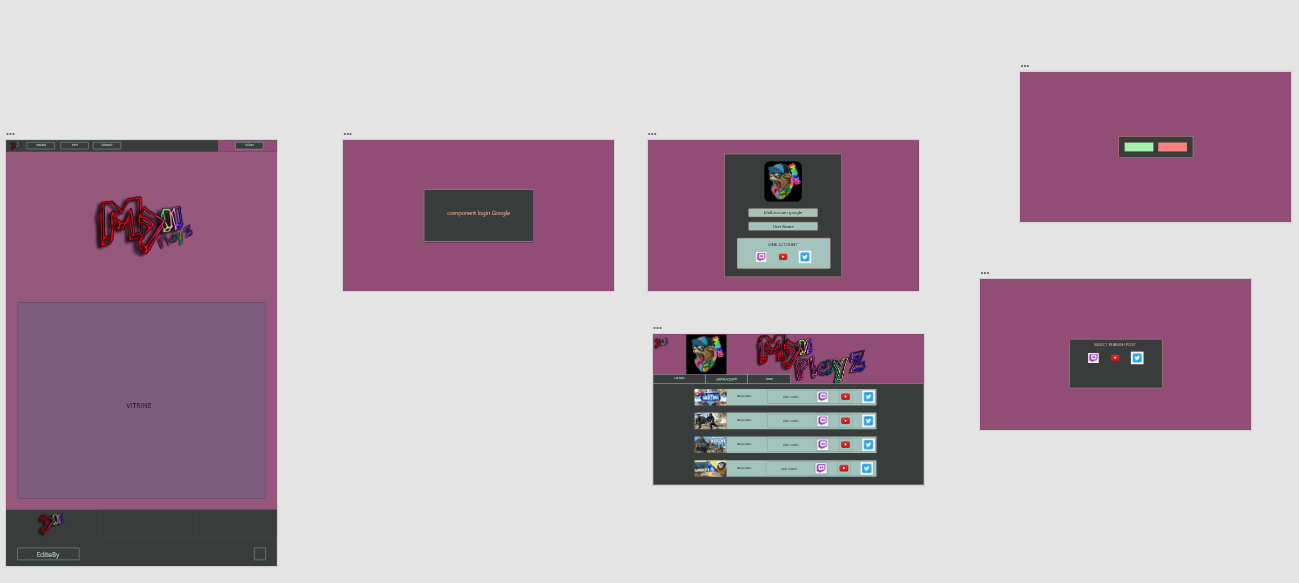
# Maquette Adobe xD

Pour l’étude UX UI nous allons nous appuyer sur une première vision du site avec une première maquette.

## Type mobile First :



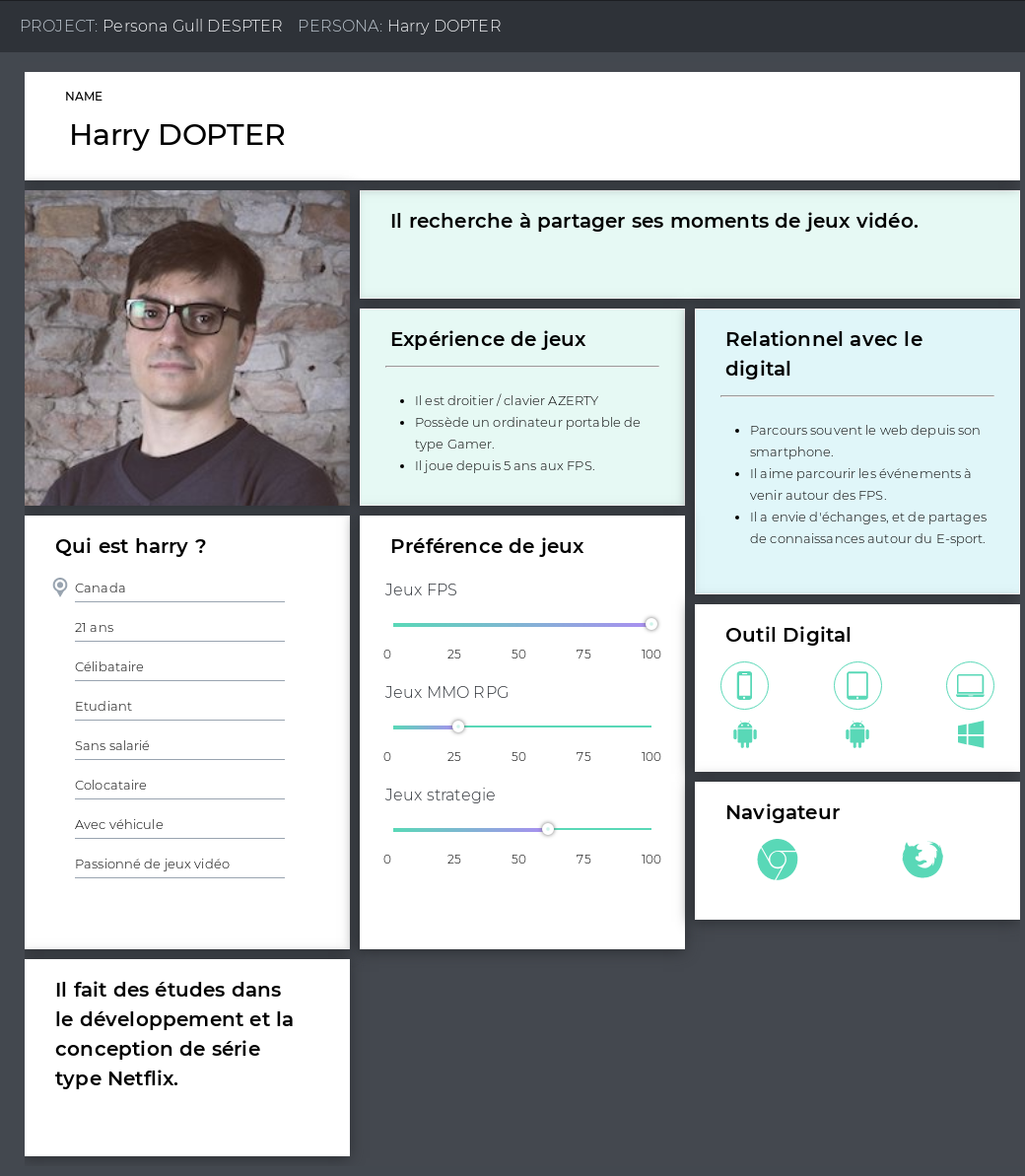
## Type desktop :



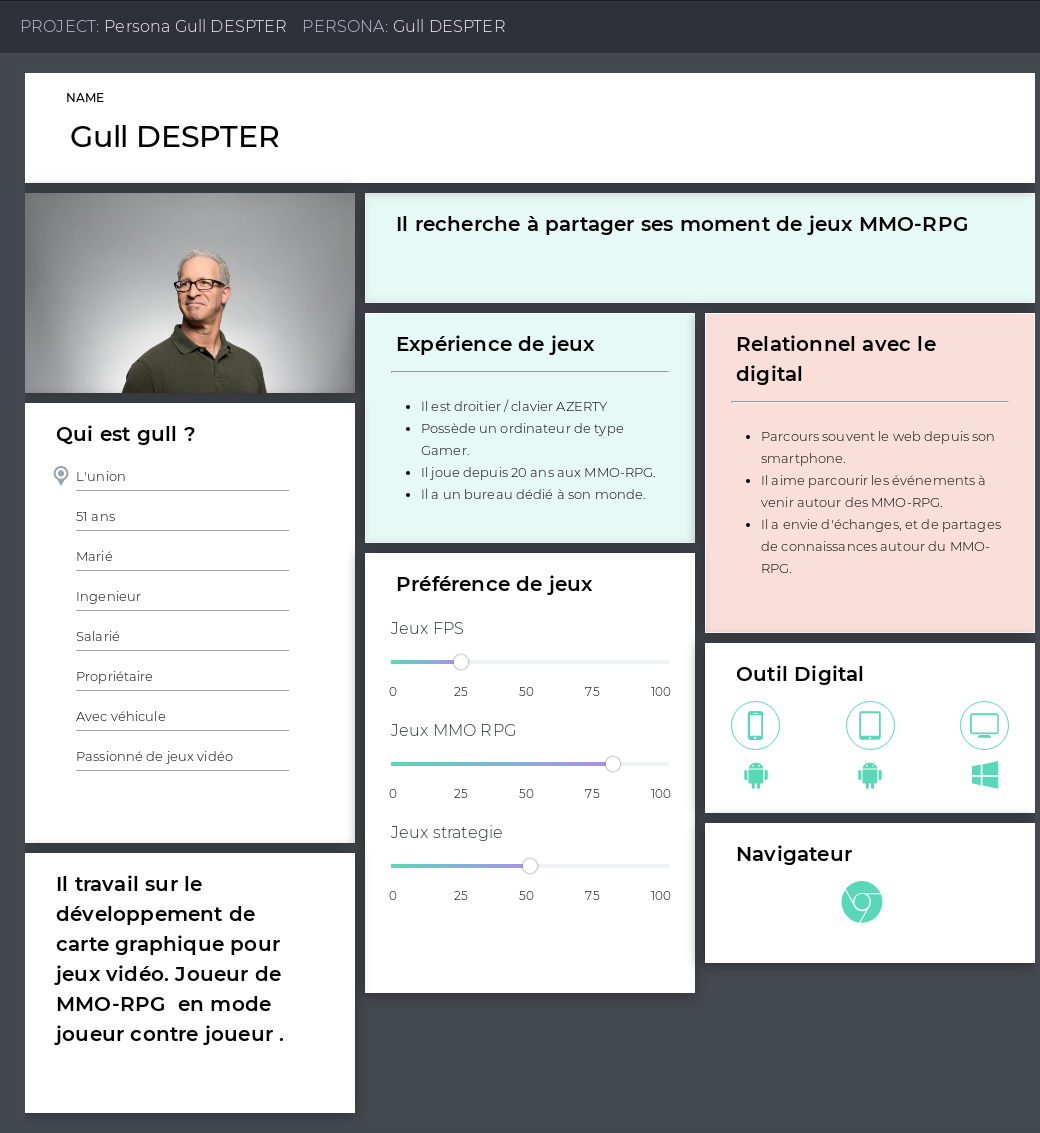
Nous allons mettre en place les personas et ainsi les faire navigué sur cette maquette affin de produire un User experience.

# Persona

## Persona Primaire



## Persona Secondaire



## Ux des utilisateurs

* Afficher l’expérience user

# Zoning des fonctionnalités

# Prototypage Bureau et Mobile Final

# Présentation base de données MongoDB

## Présentation des données :

# Modules

## React-router-dom

$ npm install --save react-router-dom

## React-toastify

$ npm install --save react-toastify

# Account management

# Mise en place de la connexion via Account-google

# Récupérations des données utilisateurs avec la connexion Gmail

# Sécurisation

## Utilisation de méthodes

Toutes les données qui arrivent via les arguments de méthode doivent être validées et les méthodes ne doivent pas renvoyer de donnée auxquelles l’utilisateur ne devrait pas avoir accès.

Nous retrouvons :

* Validation de tous les arguments avec « check » ou « npm-schema »
* Ne pas transmettre userId du client
* Une seule méthode par action
* Limitation du débit de connexion

## Les publications

Toutes les données qui arrivent via les arguments de publication doivent etre validées et les publications ne doivent pas renvoyer de donnée auxquelles l’utilisateur ne devrait pas avoir accès.

Nous retrouvons :

* Les règles relatives aux méthodes s’appliquent toujours voir au-dessus.
* Toujours restreindre les champs
* Publications renvoyée que par l’userId connecté

## Fichier servis

Nous devons assurer qu’aucun des fichiers de code source ou de configurations servis au client ne contient de données secret.

Nous retrouverons :

* La mise en place d’un code serveur secret
* La sécurisation des clés API
* Paramètre sur le cleints
* Clés API pour Oauth

## SSL

Les en-têtes http pour améliorer la sécurité des applications.

Meteor npm install helmet –save

## List de contrôle de sécurité

1. S’assurer que l’application ne dispose pas des packages ‘insecure’ ou ‘autopublish’
2. Validez tous les arguments de méthode et de publication, et inclure ‘audit-argument-checks’ pour la vérification automatique.
3. Deny dans le ‘profile’champ des documents utilisateur.
4. Utilisons des méthodes au lieu d’insérer /mettre à jour/ supprimer et ALLOW/DENY coté client.
5. Utilisé les sélecteurs spécifique et champs de filtre dans les publications
6. Ne pas utiliser raw HTML d’inclusion dans blaze à moins de bien le connaitre.
7. D’assurer que les clés API et les mots de passe secrets ne figurent pas dans le code source
8. Sécuriser les données par l’interface utilisateurs.
9. Ne pas faire confiance aux ID utilisateurs transmis par le client. UTILISEZ this.userId à l’intérieur des méthodes et des publications.
10. Configurez des en-tete http sécurises a l’aide de HELMET, mais a s’avoir que tous les ordinateur ne les prenne pas en charge. Il offre une couche supplémentaire aux utilisateurs dotés de navigateurs modernes.

# Description des principales fonctionnalités dans l’évolutions

## Mise en place de plusieurs plateformes pour une connexion au site

La mise ne place de Google connexion avec une autre plateforme-t-elle que Facebook ou tweeter serai l’idéal pour toucher le plus d’utilisateurs dans la simplicité de créations de compte et de connexions aux sites.

## Plus de plateformes de publications

Des plateformes-t-elle que Facebook ou snap sont difficile d’accès car les restrictions de publications sont juridiquement complexe mais serai un atout dans le modules vidéos short clip qui sera en futur développement.

## Gestionnaire de publication

La gestion de publications pourrai permettre de supprimé/ modifier une vidéo directement sur le site qui lui-même.

## Agenda de publication différer

Permettre aux utilisateurs de pouvoir poster des vidéos près charger sur un planning défini sur le site.

## Editeur de vidéo et outils digital

# Contraintes techniques

La mise en place du UPLOAD vers les différents prend du temps car chaque plateformes a c’est propre langage et leurs propre fonctionnements. La prospection des Api sont longs et fastidieuse.

Les limitations des plateformes, car les démarches pour les Api sont longue et des dossiers doivent être monté pour les structures de type Facebook. Afin de bien respecter les modalités de publication. Car eux même surveille beaucoup les données ententes.

Mon environnement de travail a été complètement instable sur les dernières mois donc j’ai dû suivre les cours dans un contexte compliqué et aussi de travail sur mon projet personnel, extrêmes difficile. J’ai perdu mon disque dur et aussi mon processeurs en pleine année, ce qui ne pas aider , j’ai pu récupérer mon projet sur GIT mais pour le processeurs j’ai dû rester dans la difficulté pendent pas mal de temps.

# CSS

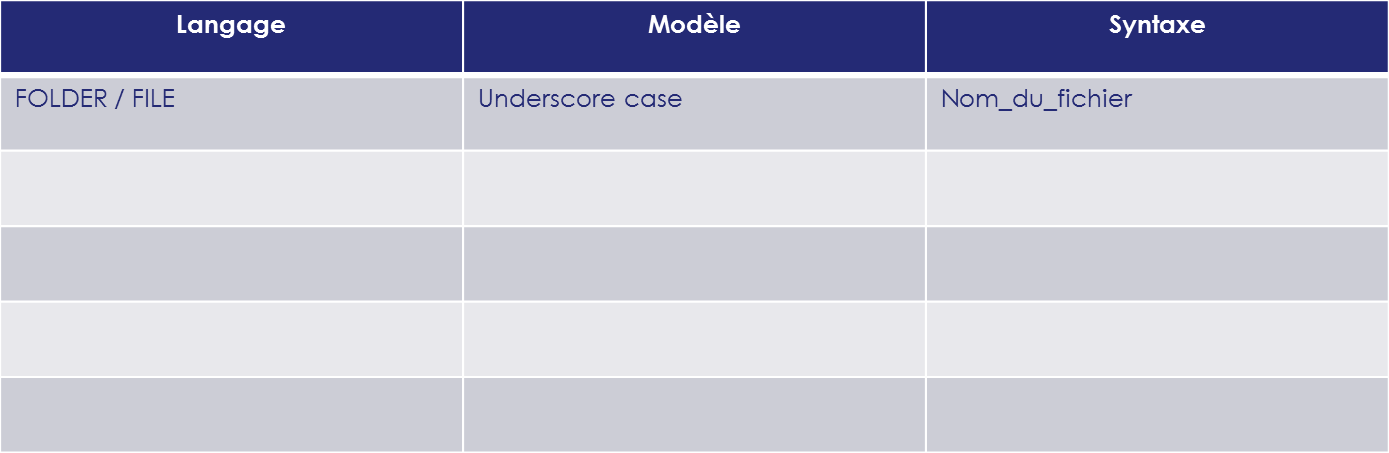
## Variable

Les propriétés personnalisées CSS ( custom properties en anglais, aussi parfois appelés variable CSS ) sont des entités définies par les développeurs ou les utilisateurs d’une page Web, contenant des valeurs spécifiques utilisables à travers le document.

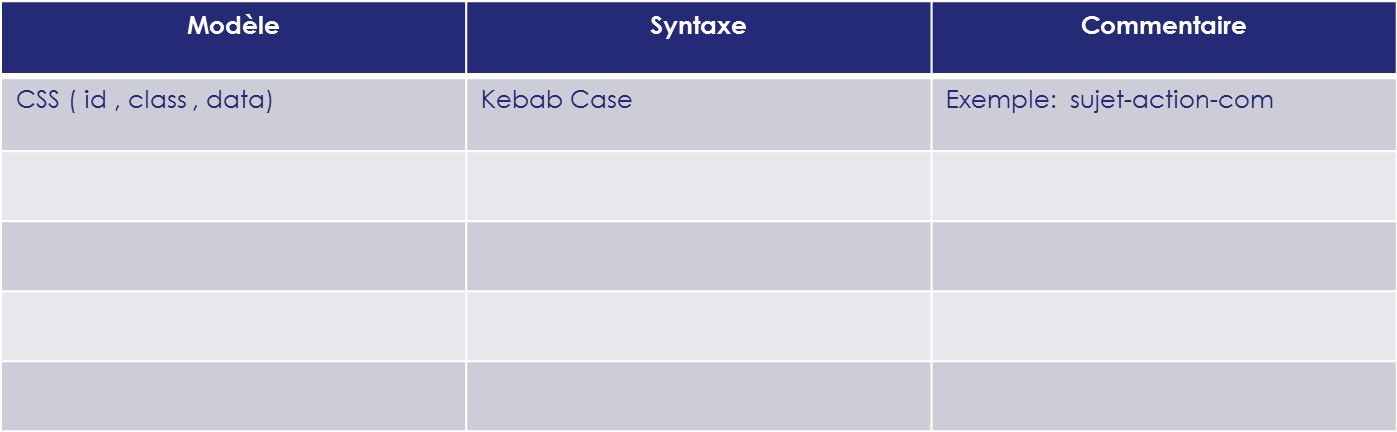
VCSS COLORS :

## Normalisation Naming

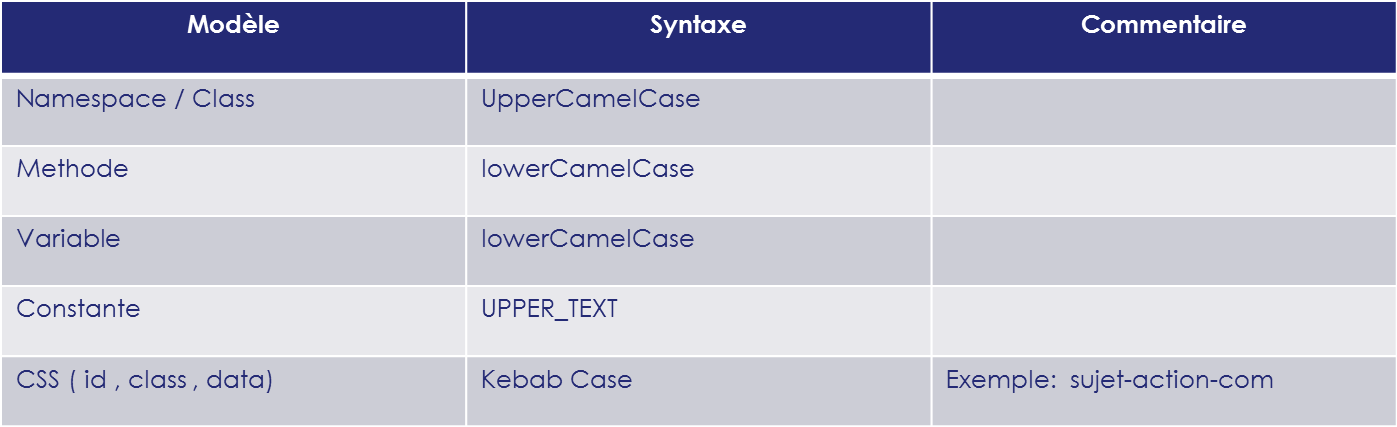
Syntaxes Global



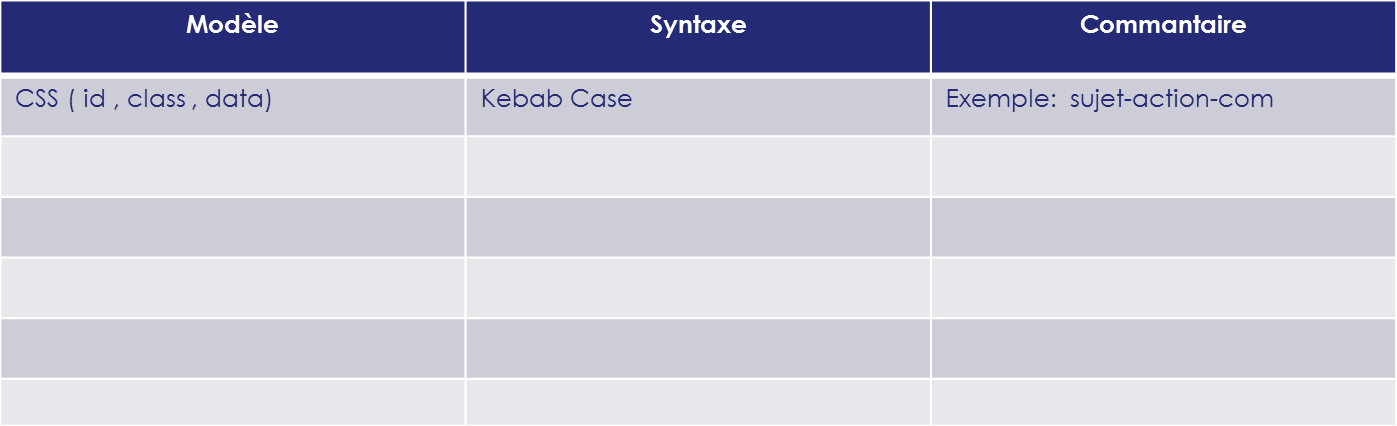
Syntaxes HTML



Syntaxes JS



syntaxes CSS



## Structure CSS

Les fichiers CSS sont spécifique à chaque page pour l’optimisation sauf les fichiers de type layout que l’on retrouve partout. Donc ils sont intégrés dans le fichier racine APP. On les reconnait à leurs underscord devant l’intitulé du fichier.

# Projection du site

# Free accès

# Forfait Premium

# Conclusion

# Remerciements

# Annexe