# 

# DOSSIER DE SYNTHESE

**Concepteur développeur web**

**Sommaire**

[**DOSSIER DE SYNTHESE** 2](#_Toc58083844)

[I. Présentation du projet 5](#_Toc58083845)

[II. Etude concurrentielle 6](#_Toc58083846)

[1. Première étude 6](#_Toc58083847)

[2. Deuxième étude 7](#_Toc58083848)

[3. Conclusion Étude concurrentielle 8](#_Toc58083849)

[II. Règles adopté de l’Ux et Ui 9](#_Toc58083850)

[III. Recherche et développement 11](#_Toc58083858)

[1. Les outils utilisés pour la recherche et le développement ont été : 11](#_Toc58083859)

[2. Liste des technologies utilisées : 11](#_Toc58083860)

[3. Tools de design : 14](#_Toc58083864)

[4. Environnement de travail : 15](#_Toc58083865)

[5. Outil de stockage des données : 15](#_Toc58083866)

[4. Choix des plateformes de partage 16](#_Toc58083867)

[5. Management avec JiraSoftware 17](#_Toc58083868)

[IV. Charte graphique 20](#_Toc58083869)

[V. Maquette Adobe xD 22](#_Toc58083874)

[Type mobile First : 22](#_Toc58083875)

[Type desktop : 23](#_Toc58083876)

[VI. Persona 24](#_Toc58083877)

[Persona Primaire 24](#_Toc58083878)

[Persona Secondaire 25](#_Toc58083879)

[Ux des utilisateurs 26](#_Toc58083880)

[VII. Zoning des fonctionnalités 27](#_Toc58083881)

[Fonctionnalité du site 27](#_Toc58083882)

[Zoning fonctionnel de l’upload 28](#_Toc58083883)

[VIII. Méthodologie de d’upload des vidéos 29](#_Toc58083884)

[1. Méthode de poste avec Twitch 29](#_Toc58083885)

[2. Méthode de poste avec Twitter 34](#_Toc58083889)

[3. Méthode de poste avec YouTube 36](#_Toc58083897)

[IX. Prototypage Bureau et Mobile final 38](#_Toc58083901)

[X. Présentation base de données MongoDB 39](#_Toc58083902)

[XI. Modules 41](#_Toc58083903)

[XII. Accounts management 43](#_Toc58083904)

[Mise en place de la connexion via Account-Google 43](#_Toc58083905)

[Récupérations des données utilisateurs avec la connexion Gmail 43](#_Toc58083906)

[XIII. Sécurisation 44](#_Toc58083907)

[1. Utilisation de méthodes 44](#_Toc58083908)

[2. Les publications 44](#_Toc58083909)

[3. Fichier servi 44](#_Toc58083910)

[4. List de contrôle de sécurité 45](#_Toc58083911)

[XIV. Contraintes techniques 46](#_Toc58083912)

[XVI. CSS 47](#_Toc58083913)

[1. Variable 47](#_Toc58083914)

[2. Normalisation Naming 48](#_Toc58083915)

[3. Structure CSS 48](#_Toc58083916)

[XV. Projection du site 49](#_Toc58083917)

[1. Description des principales fonctionnalités dans l’évolution 49](#_Toc58083918)

[2. Modération d’accès 50](#_Toc58083925)

[XVI. Conclusion 51](#_Toc58083926)

[XVII. Remerciements 52](#_Toc58083927)

[XVIII. Annexe 53](#_Toc58083928)

# Présentation du projet

Mon projet de fin d’étude est une future application côté client. Elle est développée en Javascript avec la librairie React et le framework Meteor est utilise Node.js et MongoDB comme gestionnaire de données. Cette plateforme permet à l’utilisateur de pouvoir publier des vidéos sur le thème des jeux vidéos vers des plateformes spécifiques t'elles que Twitch, Youtube, Twitter simultanément. Un espace personnel est développé autour de l’utilisateur. Pour débuter ce projet, nous utiliserons la plus simple des plateformes et la plus représentatives : Twitch

La fonctionnalité principale est de pouvoir poster une vidéo sur plusieurs réseaux aux types et format adapté à celui-ci. Un des facteurs principaux du projet est la persistance de donnée avec un hébergement indirect.

Ce projet est un tremplin vers des successions d’activités qui nous permettrons avec les accords des différents distributeurs comme Facebook, Instagram, Snapchat pour créer des Clips de publications avec des outils d’éditions.

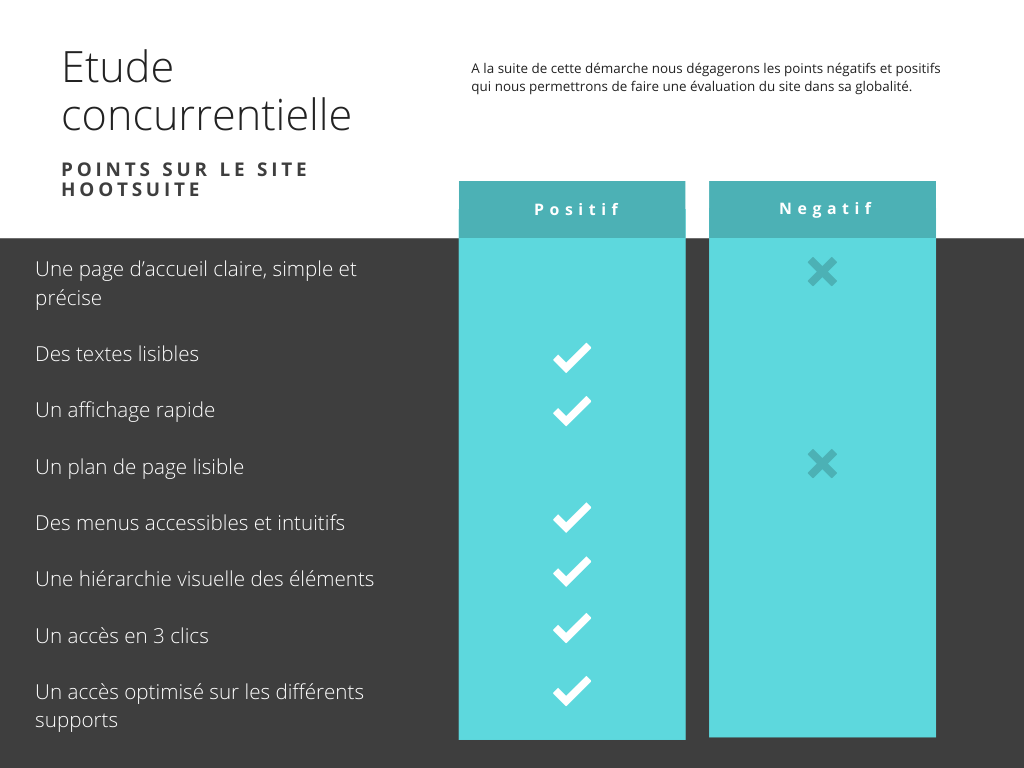
Les contraintes techniques pour ces plateformes son juridique.

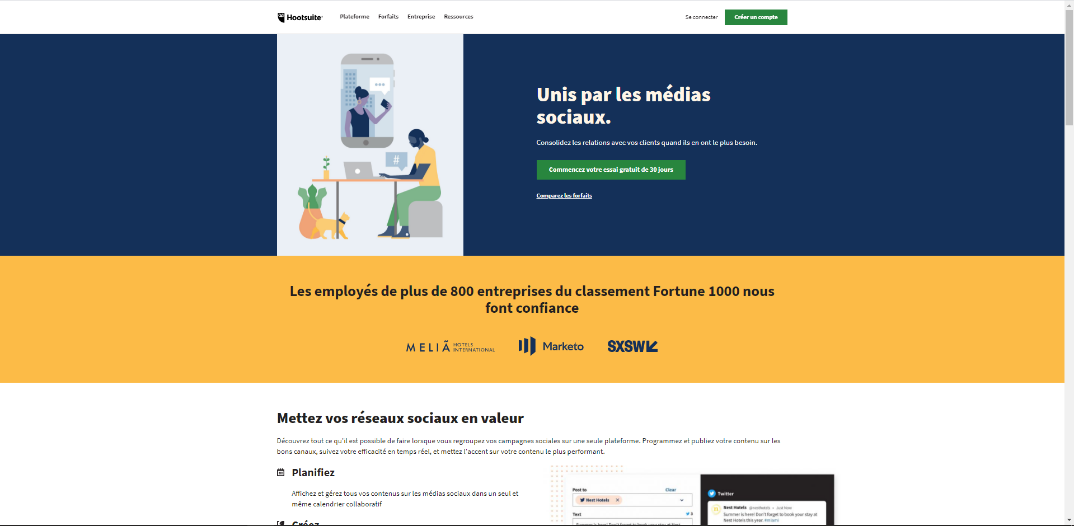
Puis avec le temps nous agrémenterons de plateforme et de fonctionnalité. Avec l’ambition de créé un outil de management de médias pour le monde professionnel et public du jeu vidéo ou des activités médiatiques qui permettre aux différents acteurs de pouvoir organisés sont plannings de publications mais aussi de les différée.

# Etude concurrentielle

## Première étude

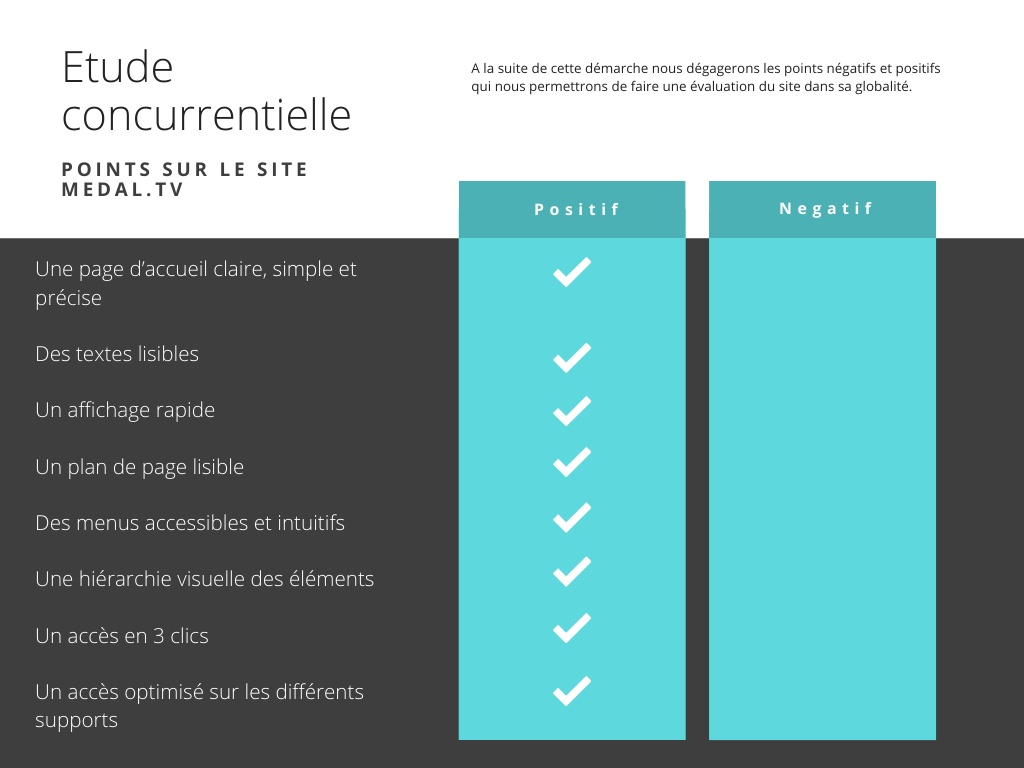
Le site web ciblé : <https://hootsuite.com/fr/>, ce site permet de prévoir les postes et l’organisation de partage de médias sur les réseaux sociaux. Son utilisation principale reste la gestion de partage de vidéo et de management de son planning de partage. Cette étude concurrentielle sur le site hootsuite a mis en évidence le besoin du partage des média simultanément sur différentes plateformes avec une grande simplicité.

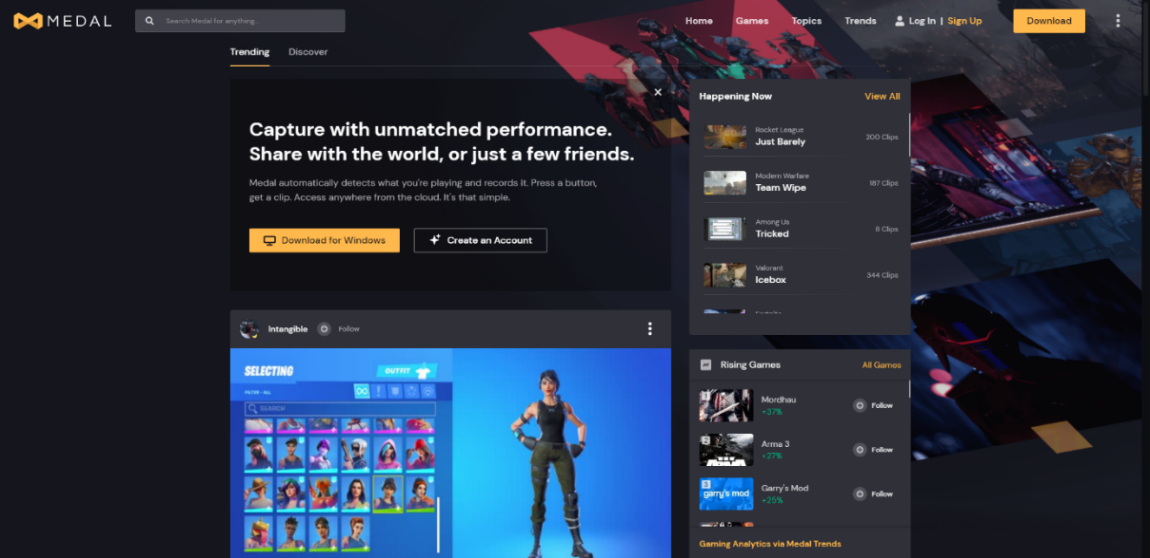




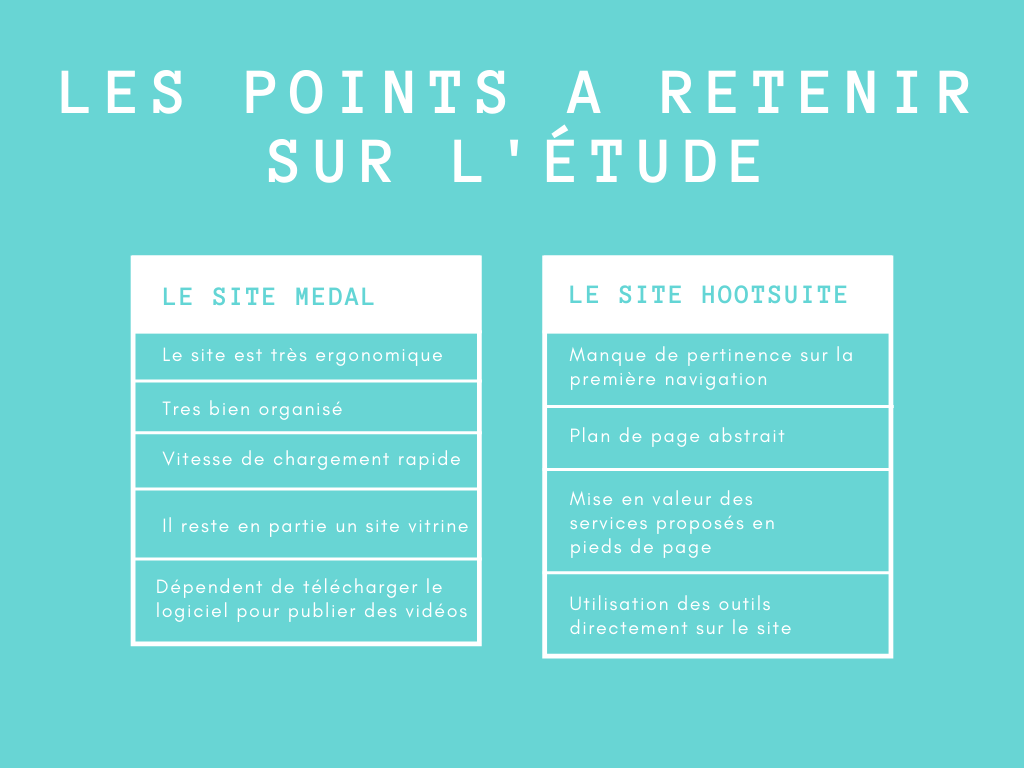
## Deuxième étude

Le site web ciblé : <https://medal.tv/> est le meilleur moyen d'enregistrer, de regarder et de partager des clips de jeu avec le plus faible impact FPS possible. Sur ce site, vous pouvez synchroniser les clips sur un cloud pour libérer de l'espace personnel sur votre ordinateur et créer votre profil. Il est possible de visualiser les clips sur le site, mais nous devons installer un logiciel pour la publication.





## Conclusion Étude concurrentielle



Les deux sites traitent des sujets bien précis et se complètent. J’ai souligné plusieurs points intéressants sur l’organisation du site MEDAL.TV en gardant les principes de Hootsuite qui est de tout avoir disponible directement sur l’application Web. Je tiens aussi à souligner la simplicité de navigation des deux sites que je garde comme ligne de conduite dans les domaines fonctionnels et visuels.

# Règles adoptées 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 mais, la simplicité de trouver une information.

Le premier objectif est de leur simplifier la vie, toutes les pages de l'application 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 … Par mon vécu, j’ai pris la décision d’utiliser une typographie spécialisée pour les dyslexiques ou les personnes avec des troubles de la lecture. Je souhaite apporter des facilités et intégrer des outils de lecture.

Voici les points à privilégier :

* 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 et sous-titres
* Chaque page doit avoir un titre et être aérée par une mise en page agréable.
* Utilisation de sous-titres, de paragraphes courts et concis.

### 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.

### Des menus accessibles et intuitifs :

Les visiteurs ne devront pas passer plus de 10 secondes à tenter de comprendre comment fonctionne le site. Les menus doivent être clairs et accessibles 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 test à suivre : flouter légèrement une page avec Photoshop et constater où 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 :

À 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/)
* JiraSoftware management

## Liste des technologies utilisées :

### **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.

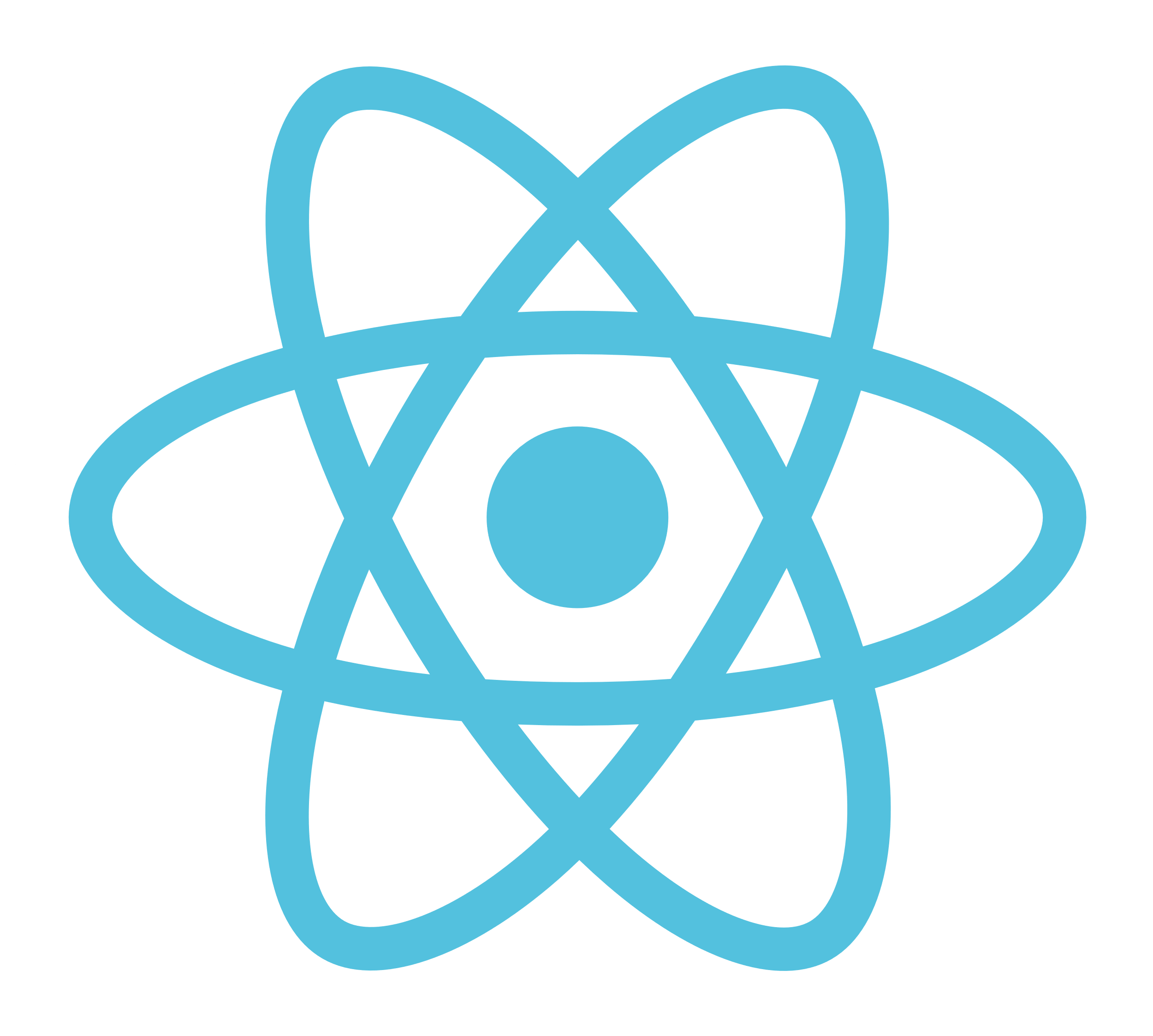
### Les langages utilisés :

 HTML5 dernière révision majeure du HTML finalisée le 28 octobre 2014.

 CSS3 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. Dans les années 2000 CSS a été pris en charge par les navigateurs Web.

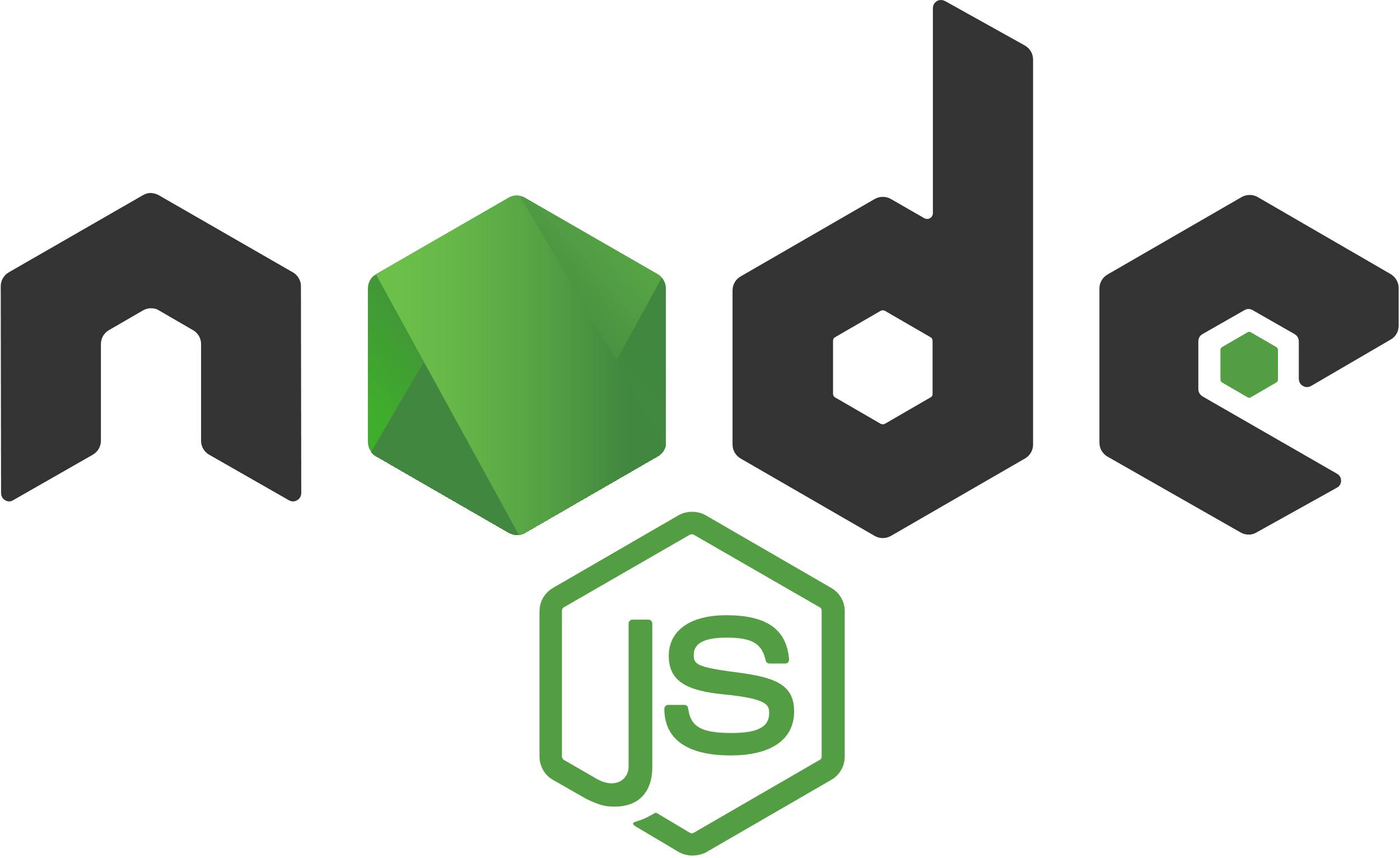
 Javascript un langage de programmation de scripts principalement employé dans les pages web interactives, mais aussi pour les serveurs avec l'utilisation de Node.js

### 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 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/Système_de_gestion_de_base_de_données) 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_réactive) 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.

#### Avantage de React, Meteor, Node.js

DOM (document Object model) est un compromis des vues 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. En finalité, 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 Framework 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.

## 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 mûrir les idées avec cet outil et j'ai pu commencer à intégrer des items graphiques.

** Adobe xDest 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 compatibles avec Google que j’ai utilisé afin de mettre en place le zonning des fonctionnalités.

## Environnement de travail :

* Mon ordinateur personnel contient :

➥ **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 Réseau Orange fibre (800mb/s débit descendent 300mb/s débit montant).

## Outil de stockage des données :

Base de données MongoDB dont les appels sont faits et gérés par Meteor.

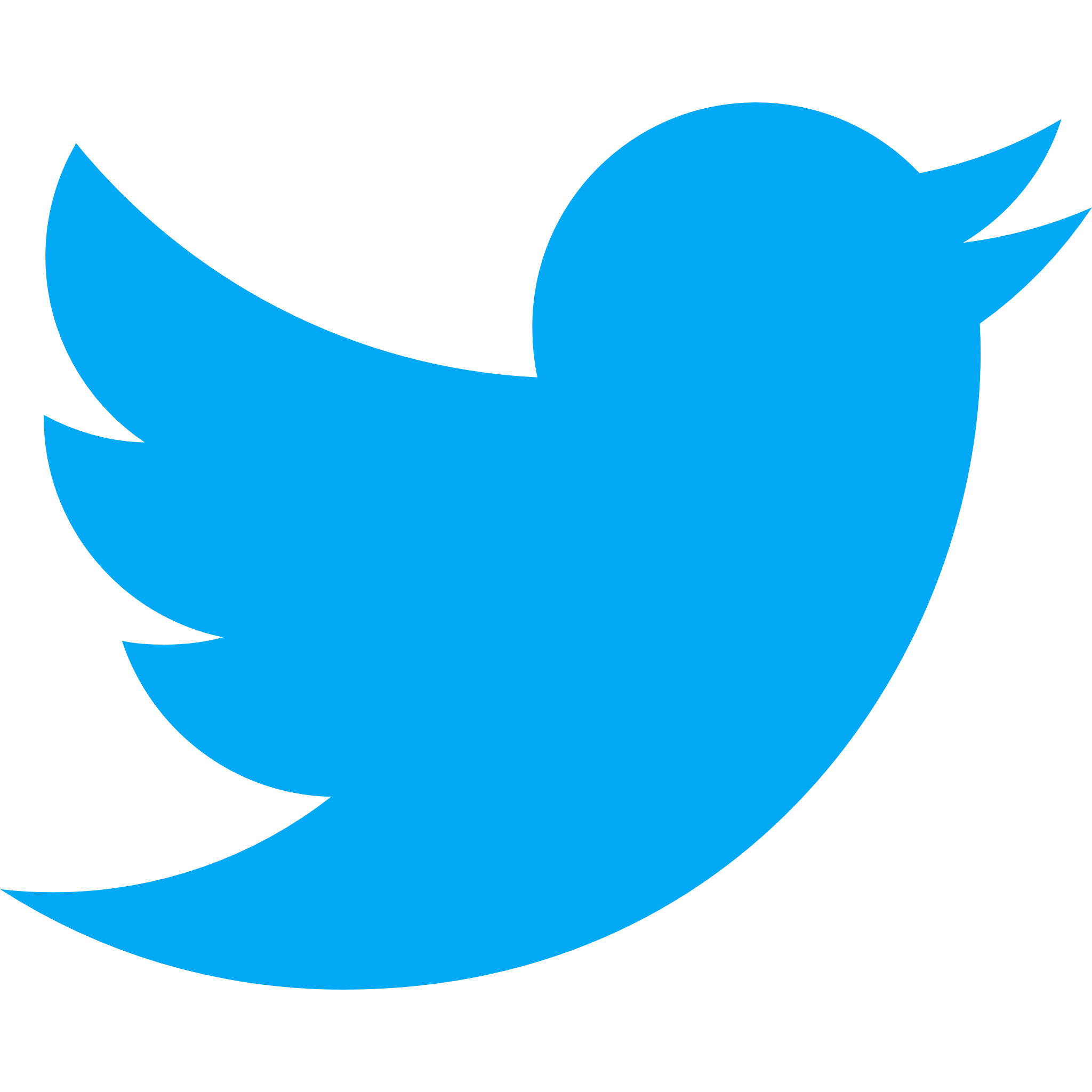


Héberger chez qui propose plusieurs services comme :

* Déploiement de code instantané.
* Tous les langages sont supportés pour les projets web.
* Bases de données à la demande. Provisionnez MongoDB en un clic.

## Choix des plateformes de partage

Twitch sera mon choix par défaut, l’application utilise un build React et une API d’upload est disponible sur le site de développement avec des requêtes URL de type JSON. Il est donc possible de pouvoir une fois connecté (OAuth) et pouvoir créer une vidéo, de la modifier ou d'en supprimer une.

Twitter est aussi une plateforme build sous React. Il est possible aussi d’upload des médias via une API avec le même format de requête URL de type JSON.

YouTube est une plateforme qui utilise un procédé de scripte Python2 ou Python3 avec des requêtes URL de format JSON.

## Management



Jira Software est un outil de gestion de projet Agile qui prend en charge toute méthodologie Agile, qu'il s'agisse de Scrum, Kanban. Grâce à un seul outil, nous pouvons : planifier, suivre, gérer nos projets de développement et tableaux aux rapports.

Les réunions de planification de sprints définissent les tâches d’une équipe qu’elle doit accomplir dans le sprint à venir, qui sont consignées dans le backlog, où répertorient toutes les tâches à réaliser. Jira Software place notre backlog au centre des réunions de planification de sprint.

#### Gestion des versions

Nous pouvons suivre les versions, les fonctionnalités et la progression en un coup d'œil. Cliquer sur une version pour consulter l'état complet, notamment les tickets, les données de développement et les éventuels problèmes.

#### Préparation du backlog en toute simplicité

On redéfinit la priorité des user stories et bugs en toute facilité. En un ou plusieurs tickets, puis glissez les dans le backlog pour les réorganiser. Mais aussi créer des filtres rapides pour remonter les tickets affichant des attributs importants.

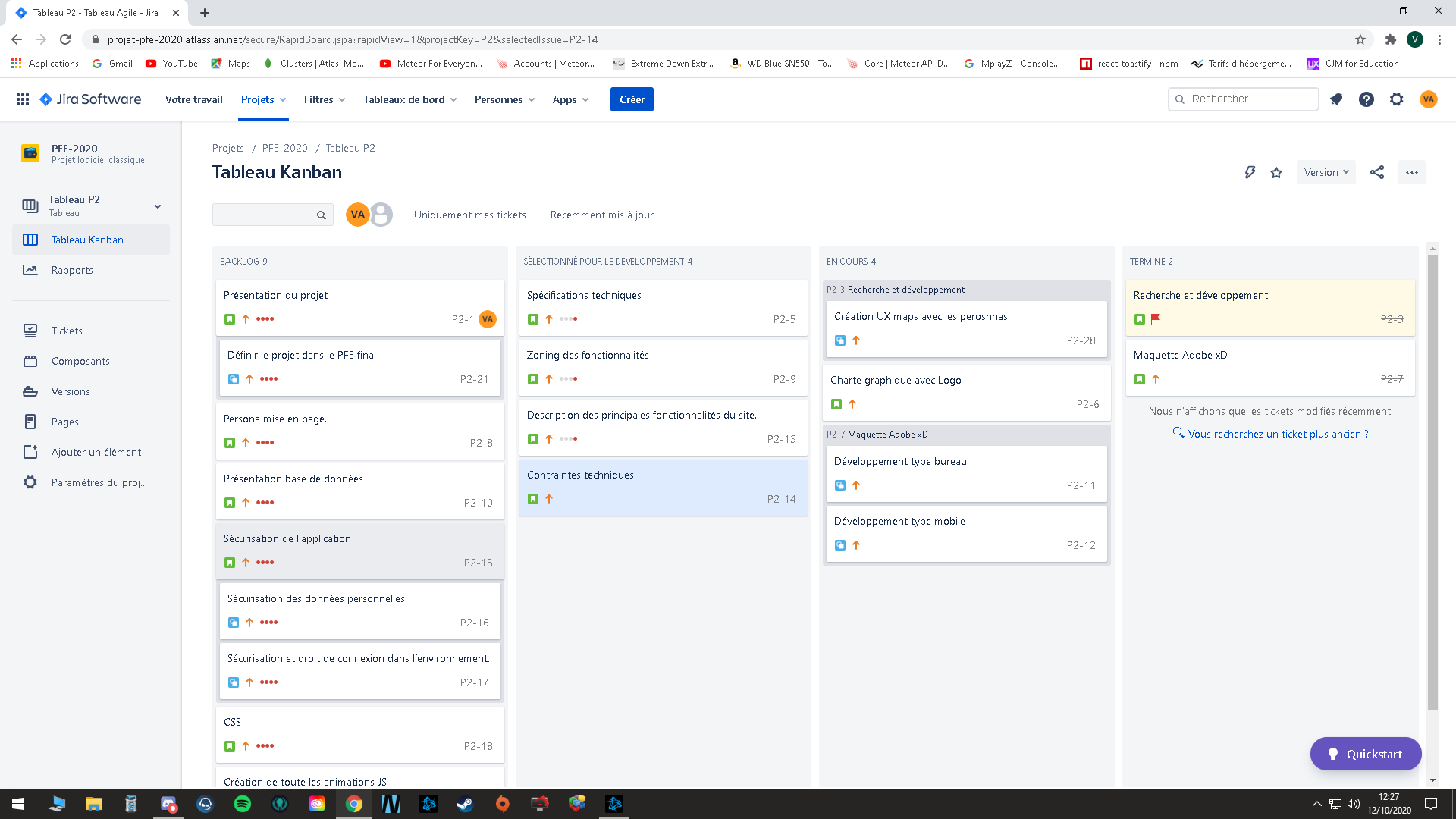
#### Planification du sprint

Il est possible de placer le backlog au centre des réunions de planification de sprint. Avec le reste d’une équipe, faire des estimations des stories, ajuster le périmètre du sprint, vérifier la vélocité et redéfinisse la priorité des tickets en temps réel.

#### Story points

Effectuer des estimations et le suivi, générer des rapports sur les story points afin qu’une équipe puisse gagner en précision dans les sprints futurs. Utiliser des story points, des heures optimales ou leurs propres méthodes d'estimation.

Voici le contexte global de mon utilisation au sein du développement de mon projet.

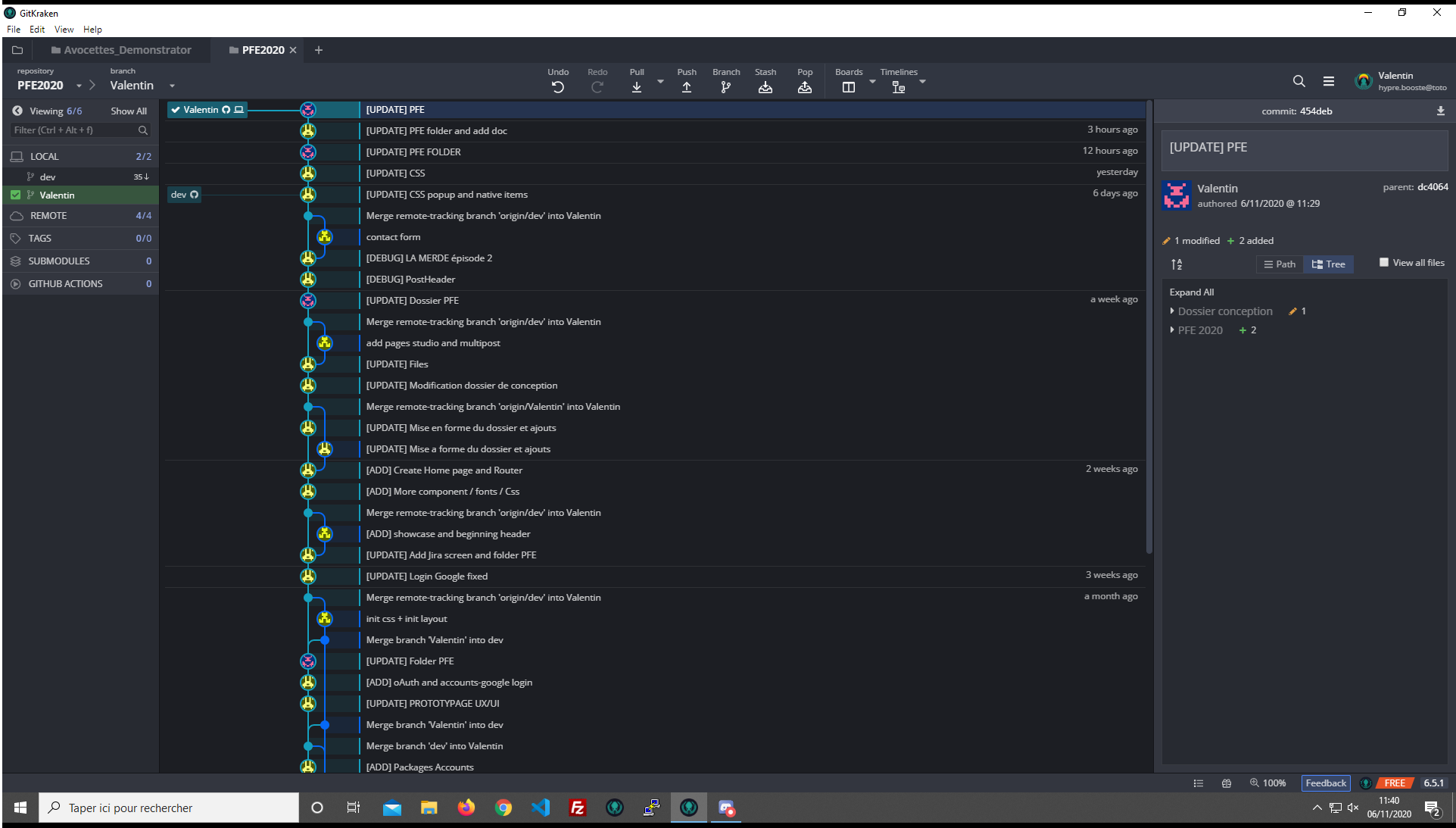


#### Git

Est un logiciel de gestion de versions (Version Control System) qui suit l'évolution des fichiers sources et garde les anciennes versions de chacun d'eux sans rien écraser. ... Il fournit une interface visuelle pour gérer localement des projets avec les contrôles de version.

#### Gitkraken

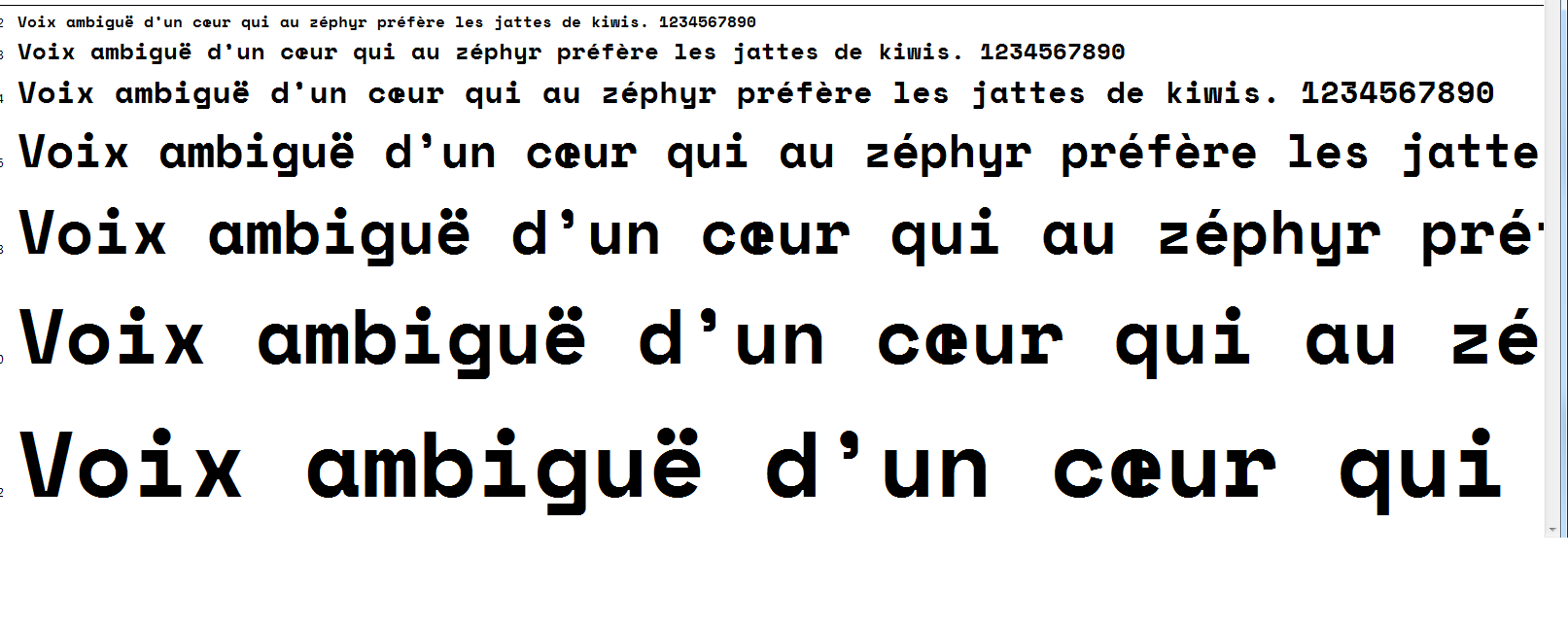
J’ai utilisé un client graphique pour Git.



# Charte graphique

### Polices / Fonts

La police intégrée dans le projet est de Google font « Monospace »



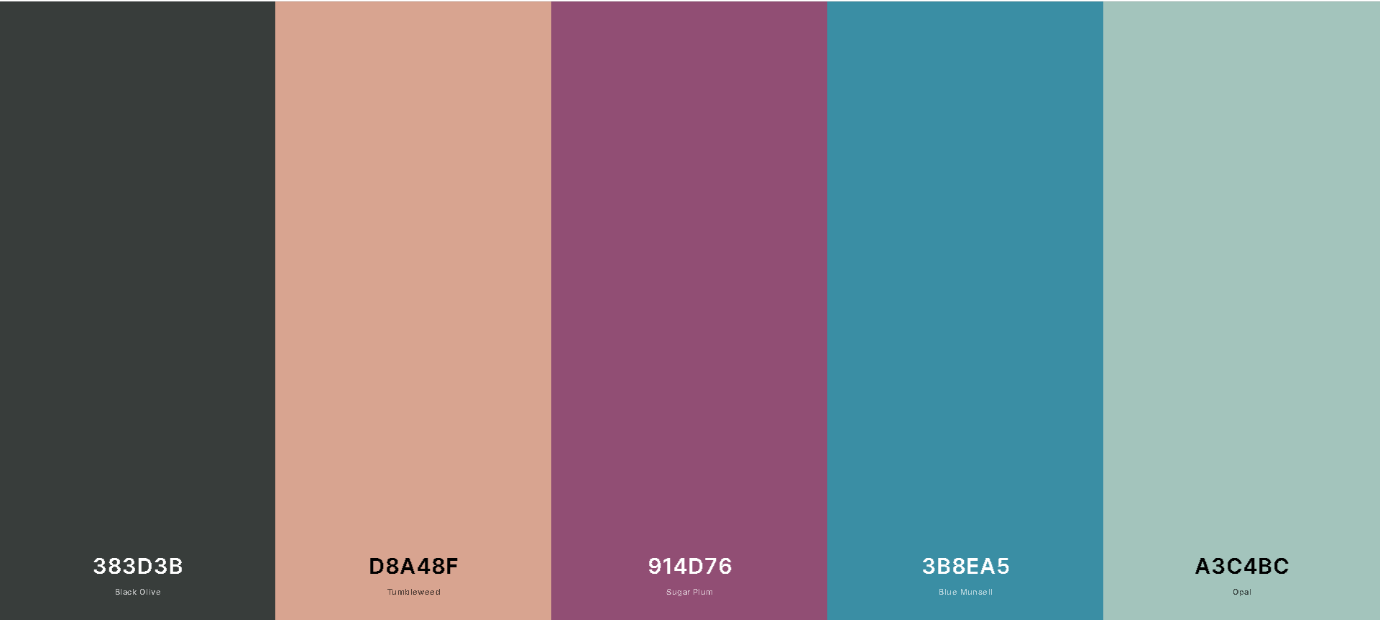
### Tailles natives

Les tailles natives du projet sont sur des chiffres ronds.

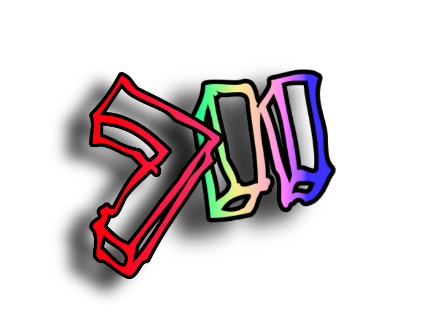
Dans le cas des références en REM elles seront présentées de 0.5 en 0.5

### Couleurs

 Les couleurs choisies sur coolors.co avec un algorithme pour le choix et les contraste.



### Logos

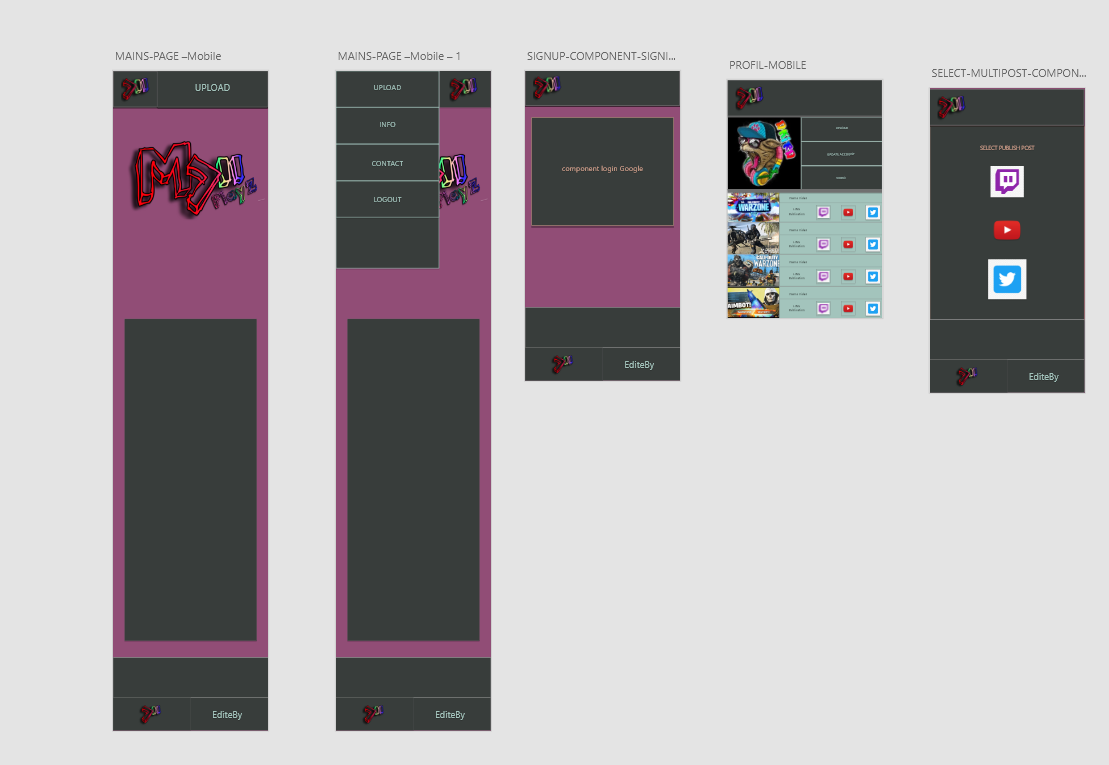




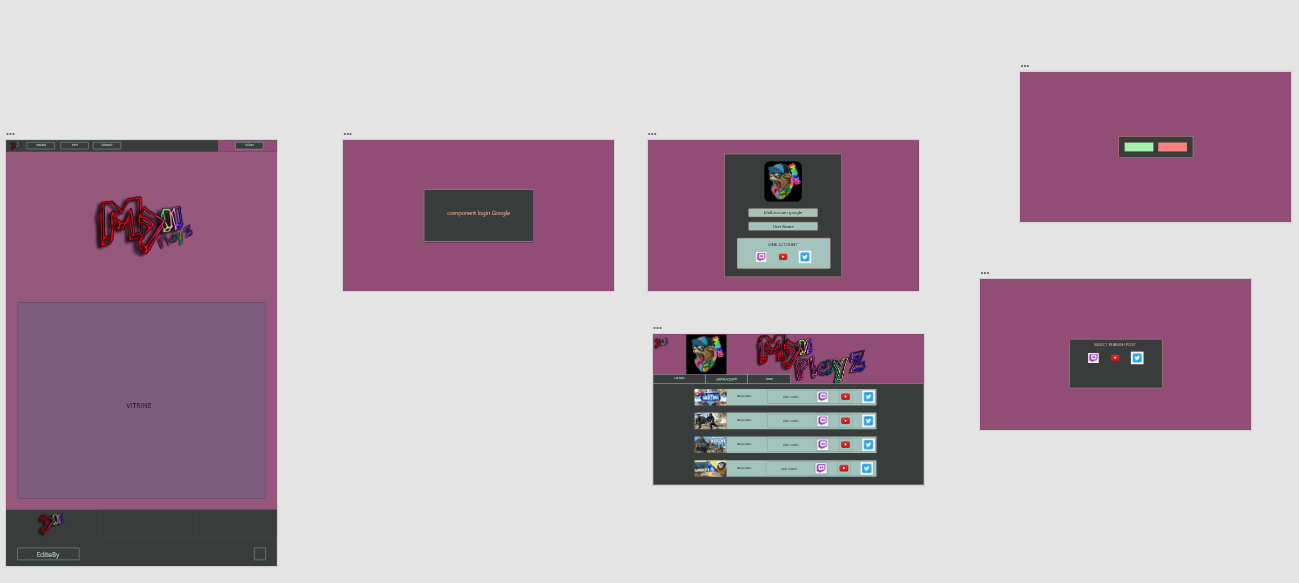
# Maquette Adobe xD

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

### Type mobile First :



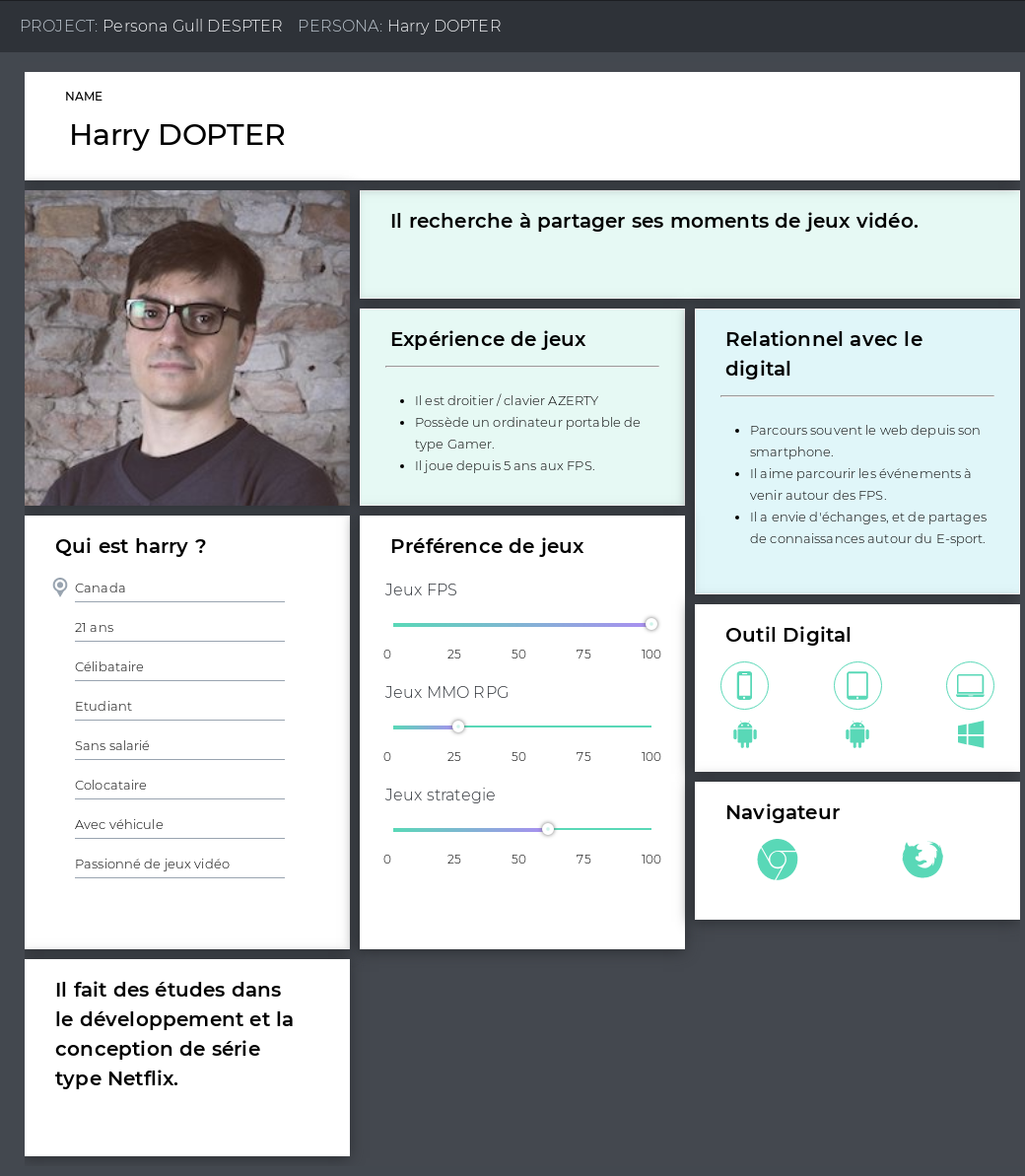
### Type desktop :



Nous allons mettre en place les persona et ainsi les faire naviguer sur cette maquette afin de produire un User expérience.

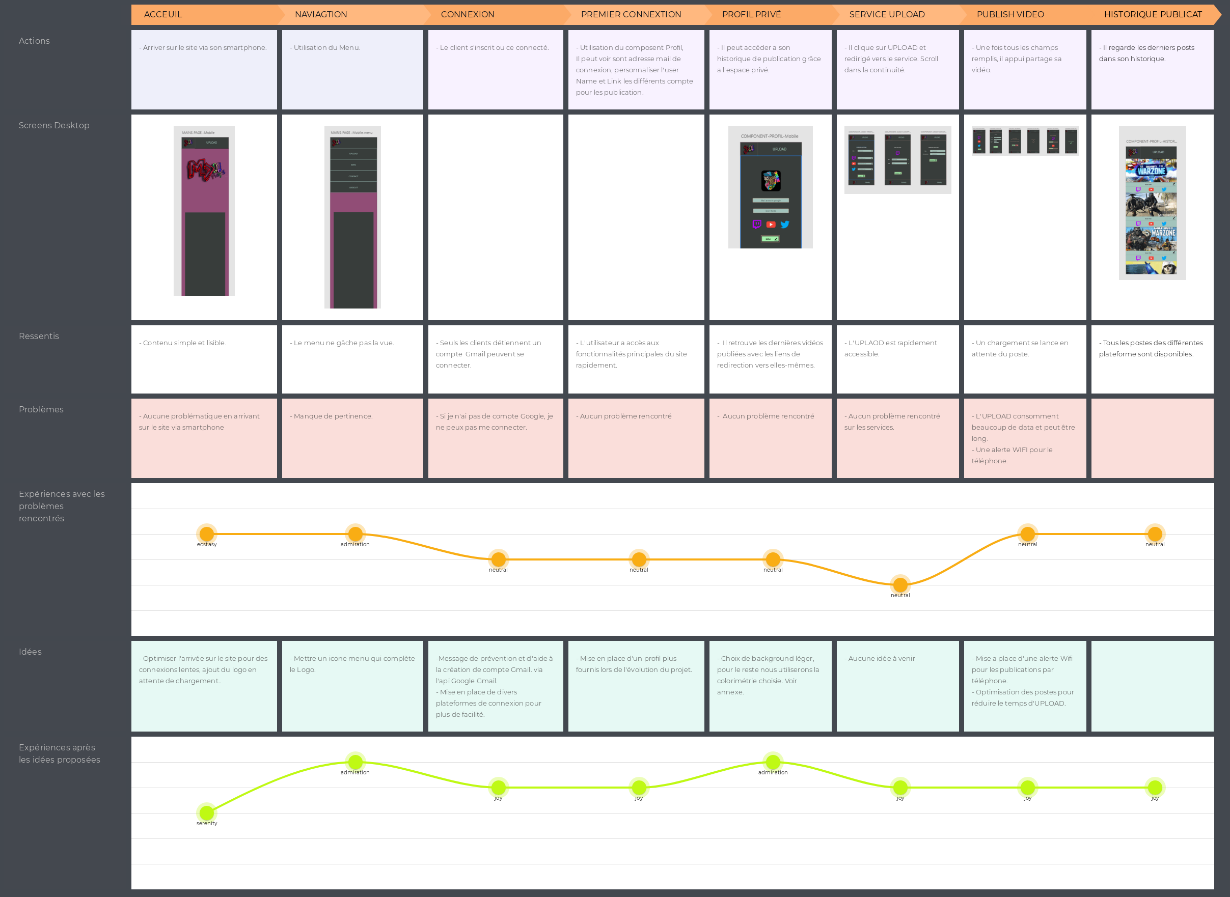
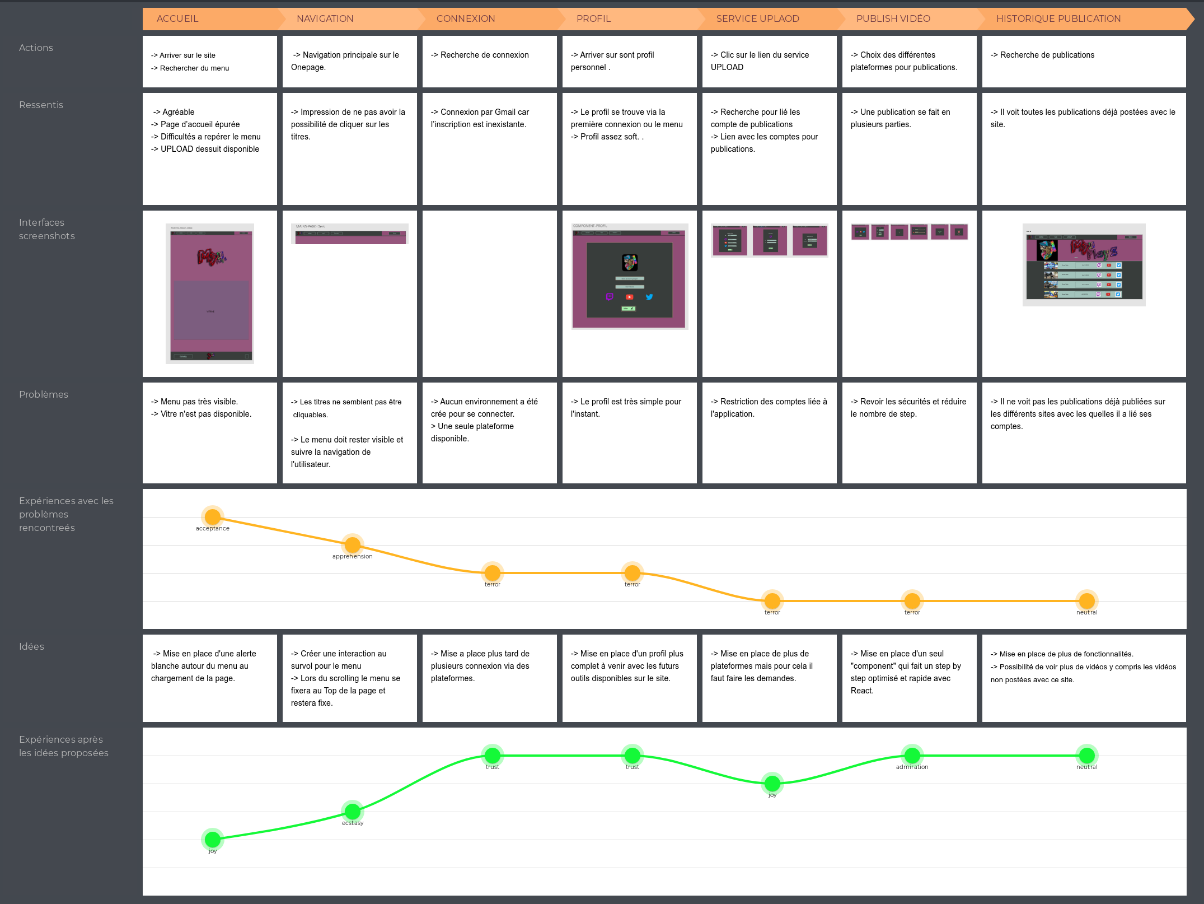
# Persona

### Persona Primaire



### Persona Secondaire

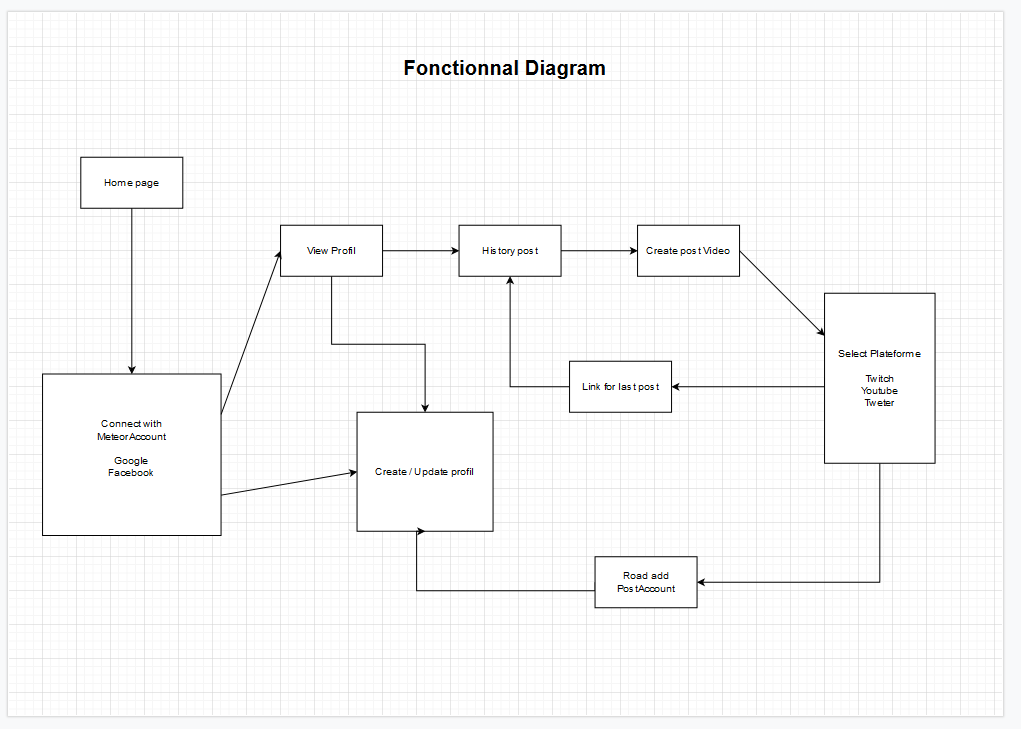
### Ux des utilisateurs



# Zoning des fonctionnalités

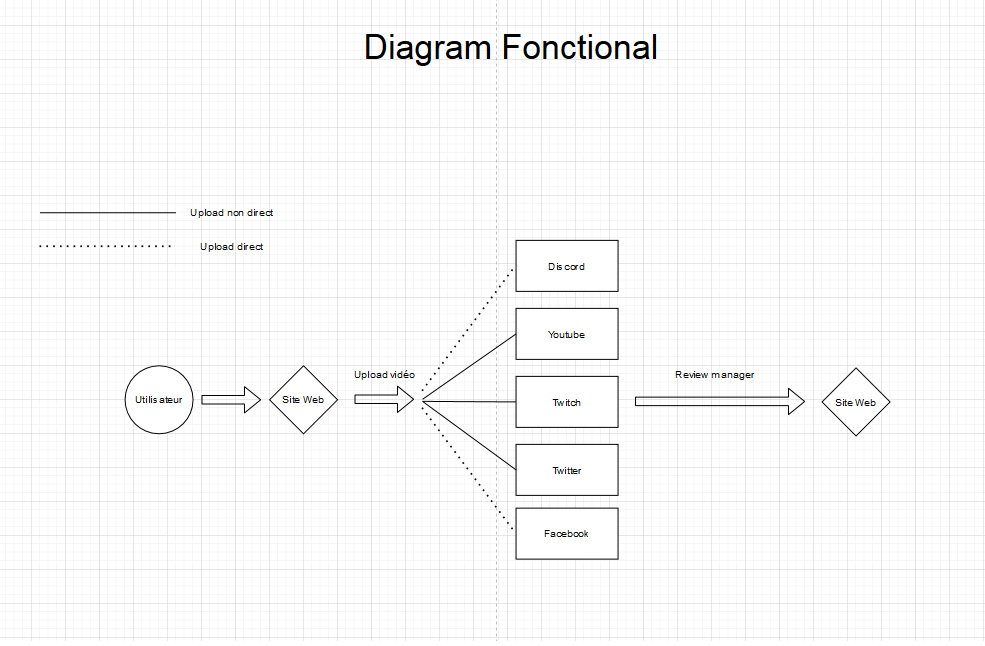
### 

### Fonctionnalité du site



### 

### Zoning fonctionnel de l’upload



# Méthodologie de d’upload des vidéos

## Méthode de poste avec Twitch

Nous allons faire appel à une API. Dans cette API les formats suivants sont acceptés :

* Formats de fichiers MP4, MOV, AVI et FLV
* Audio AAC
* Codec h264
* Débit binaire jusqu’à 10Mbps
* Jusqu’à 1080p / 60FPS

Une limite de taux de 5 UPLOAD simultanés par utilisateur avec un maximum de 100 téléchargements en 24 h.

Tout d’abord, le premier step est de pouvoir se connecter à l’API avec OAuth.

Faire appel à l’API, pour cela, nous avons besoin d’un ID client. Pour en recevoir un, nous devons nous connecter à la console des développeurs Twitch. Nous utiliserons l’accès URI de redirection avec OAuth. Une fois identifié, nous recevons un Tokken d’identification, que nous allons réutiliser pour une requête de type :

### URL POST

https://api.twitch.tv/kraken/videos?channel\_id=<channel ID>&title=<video title>

Exemple de requête :

curl -H 'Accept: application/vnd.twitchtv.v5+json' \

-H 'Authorization: OAuth cfabdegwdoklmawdzdo98xt2fo512y' \

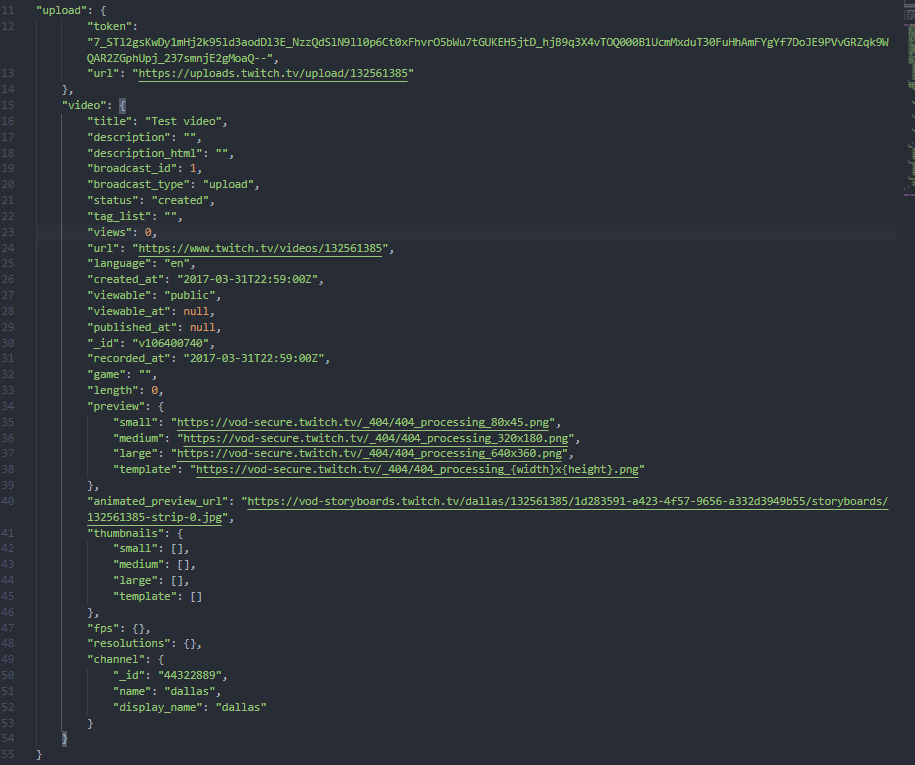
-H 'Client-ID: uo6dggojyb8d6soh92zknwmi5ej1q2' \

-X POST 'https://api.twitch.tv/kraken/videos?channel\_id=44322889&title=Test video')

Typage et description de donnée :

| **Nom** | **Type** | **La description** |
| --- | --- | --- |
| *description* | String | Brève description de la vidéo. |
| *game* | String | Nom du jeu dans la vidéo. |
| *language* | String | Langue de la vidéo (par exemple, en). |
| *tag\_list* | String | Liste de balises séparées par des virgules décrivant la vidéo (par exemple, « avions, effrayant »). Maximum: 100 caractères par balise, 500 caractères pour toute la liste |
| *viewable* | String | Spécifie qui peut voir la vidéo. Valeurs valides: public(la vidéo est visible par tout le monde) ou private(la vidéo n'est visible que par le propriétaire et les éditeurs de la chaîne). Par défaut: public. |
| *viewable\_at* | RFC3339 date | Date à laquelle la vidéo deviendra publique. Cela ne prend effet que si viewable=private. |

Format UPLOAD de type JSON :



Procédé de mise à jour les informations sur la vidéo qui ont été créée.

Tous d’abord l'authentification est nécessaire avec OAuth 2.0

Scope requis visé : channel\_editor

### URL PUT

https://api.twitch.tv/kraken/videos/<video ID>

Les paramètres de requête facultatifs :

| **Nom** | **Type** | **La description** |
| --- | --- | --- |
| *description* | String | Brève description de la vidéo. |
| *game* | String | Nom du jeu dans la vidéo. |
| *language* | String | Langue de la vidéo (par exemple, en). |
| *tag\_list* | String | Liste de balises séparées par des virgules décrivant la vidéo (par exemple, « avions, effrayant »). Maximum : 100 caractères par balise, 500 caractères pour toute la liste. |
| *title* | String | Titre de la vidéo. 100 caractères maximum. |

Exemple de requête :

curl -H 'Accept: application/vnd.twitchtv.v5+json' \

-H 'Authorization: OAuth cfabdegwdoklmawdzdo98xt2fo512y' \

-H 'Client-ID: uo6dggojyb8d6soh92zknwmi5ej1q2' \

-X PUT 'https://api.twitch.tv/kraken/videos/106400740?title=Updated test video'

Procédé de suppression d’une vidéo spécifiée :

La vidéo peut être de n’importe quel type.

L'authentification est nécessaire avec OAuth 2.0

Scope requis visé : channel\_editor

### URL DELETE

https://api.twitch.tv/kraken/videos/<video ID>

Aucun paramètres facultatifs, mise en place d’une validation « self DELETE »

Exemple de requête :

curl -H 'Accept: application/vnd.twitchtv.v5+json' \

-H 'Authorization: OAuth cfabdegwdoklmawdzdo98xt2fo512y' \

-H 'Client-ID: uo6dggojyb8d6soh92zknwmi5ej1q2' \

-X DELETE 'https://api.twitch.tv/kraken/videos/106400740'

Réponse attendue :

200 OK

{"ok":"true"}

## Méthode de poste avec Twitter

### Les formats

Les vidéos Twitter doivent répondre aux exigences minimales de la plate-forme de médias sociaux.

* La limite de taille de la vidéo Twitter est de 512 Mo (asynchrone) / 15 Mo (synchronisation)
* La résolution vidéo minimale sur Twitter est de 32 x 32
* La résolution vidéo maximale sur Twitter est de 1 920 x 1 200 et 1 200 x 1 900
* Les vidéos Twitter doivent respecter les proportions suivantes : 1 : 2,39 - 2,39: 1
* La fréquence d'images maximale pour les vidéos est de 40 ips (images par seconde)
* Le débit binaire maximal est de 25 Mb/s (mégabits par seconde)
* Twitter prend actuellement en charge les formats vidéo suivants : formats MOV et MP4 pour les applications mobiles
* La durée minimale de la vidéo Twitter est de 0,5 secondes et la durée maximale de la vidéo Twitter est de 2 minutes 20 secondes (140 secondes)
* Twitter prend en charge les formats vidéo Web suivants : téléchargements vidéo MP4 au format H264 et audio AAC.

### Requête

Les requêtes doivent être au format multipart/form-data ou application/x-www-form-urlencoded POST.

Le domaine de ce point de terminaison est upload.twitter.com

### URL de la ressource

<https://upload.twitter.com/1.1/media/upload.json>

### Informations sur les ressources

|  |  |
| --- | --- |
| **Formats de réponse** | **JSON** |
| Nécessite une authentification ? | Oui (contexte utilisateur uniquement) |
| Rate limited ? | Oui |

### Paramètres

| **Nom** | **Obligatoire** | **La description** |
| --- | --- | --- |
| *Médias* | Obligatoire | Le contenu du fichier binaire brut en cours de téléchargement. Ne peut pas être utilisé avec media\_data. |
| *media\_category* | Optionnel | La catégorie qui représente la façon dont le média sera utilisé. Ce champ est obligatoire lors de l'utilisation du média avec l'API Ads |
|  |  |  |

### POST

https://upload.twitter.com/1.1/media/upload.json?media\_category=tweet\_video

### Exemple de réponse

{

"media\_id": 710511363345354753,

"media\_id\_string": "710511363345354753",

"media\_key": "3\_710511363345354753",

"size": 11065,

"expires\_after\_secs": 86400,

"video": {

"video\_type": "video/mp4",

"w": 800,

"h": 320

}

}

## Méthode de poste avec YouTube

### Les formats

Cette méthode prend en charge le téléchargement multimédia. Les fichiers téléchargés doivent respecter ces contraintes :

* Taille maximale du fichier : 128 Go
* Types MIME : Médias video/\* compatibles ,application/octet-stream

### Mise en place de l’autorisation de connexion

Mise en place de OAuth 2.0 pour pouvoir accéder à l’API de développement.

### Vidéos

Les requetés pour les différentes actions possibles présentées par la documentation YouTube.

| **Méthode** | **Requête HTTP** | **La description** |
| --- | --- | --- |
| URI relatif à https://www.googleapis.com/youtube/v3/ | | |
| *insert* | POST /videos | Télécharge une vidéo sur YouTube et définis éventuellement les métadonnées de la vidéo. |
| *list* | GET /videos | Renvoie une liste de vidéos correspondant aux paramètres de la requête API. |
| *delete* | DELETE /videos | Supprime une vidéo YouTube. |
| *update* | PUT /videos | Mets à jour les métadonnées d'une vidéo. |

Les ressources de représentation sont grandes, mais sont disponibles sur la documentation :

https://developers.google.com/youtube/v3/docs/videos#methods

Exemple de requête :

### POST

https://youtube.googleapis.com/youtube/v3/videos?part=snippet%2Cstatus&key=[YOUR\_API\_KEY]

Exemple de data :

{

  "kind": "youtube#video",

  "etag": etag,

  "id": string,

  "snippet": {

    "publishedAt": datetime,

    "channelId": string,

    "title": string,

    "description": string,

    "thumbnails": {

      (key): {

        "url": string,

        "width": unsigned integer,

        "height": unsigned integer

      }

    },

    "channelTitle": string,

    "tags": [

      string

    ],

    "categoryId": string,

    "liveBroadcastContent": string,

    "defaultLanguage": string,

    "localized": {

      "title": string,

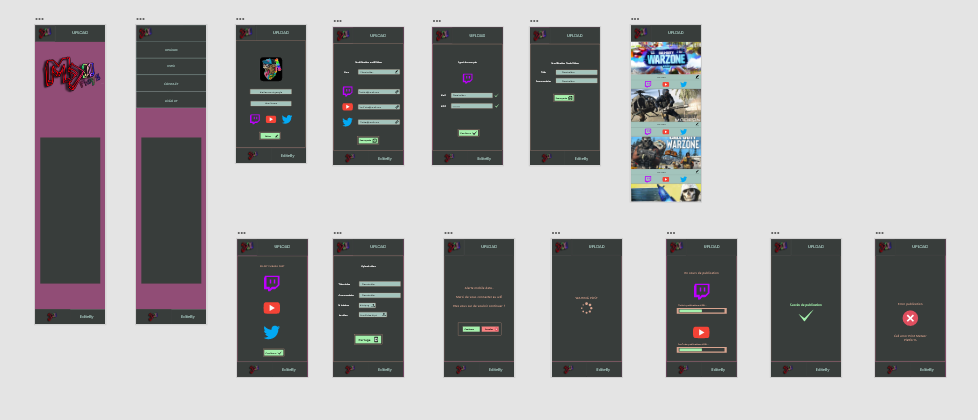
      "description": string

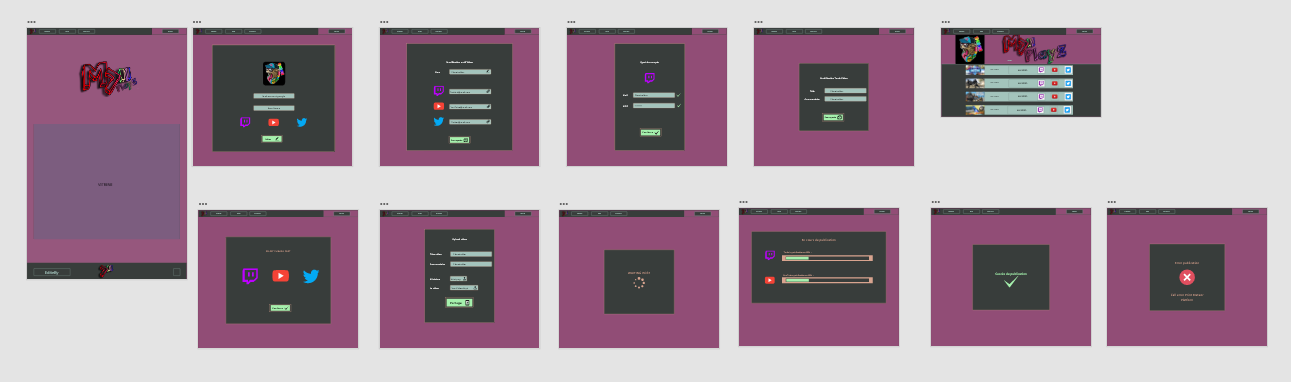
    },

    "defaultAudioLanguage": string

  },

# Prototypage Bureau et Mobile final





# Présentation base de données MongoDB

#### Modèle de données pour un utilisateur :

{

  username: {

    type: String,

  },

  services: {

    type: Object,

    blackbox: true

  },

  emails: {

    type: Array,

    optional: true

  },

  "emails.$": {

    type: Object,

    optional: true,

    blackbox: true

  },

  coloreyes:{

    type: String,

    optional: true

  },

  createdAt: Date,

}

# React hooks

Les Hooks sont des fonctions qui permettent de « se brancher » sur la gestion d’état local et de cycle de vie de React depuis des fonctions composants. Les Hooks ne fonctionnent pas dans des classes : ils nous permettent d’utiliser React sans classes.

Voici un exemple d’utilisation dans mon projet :



# Modules

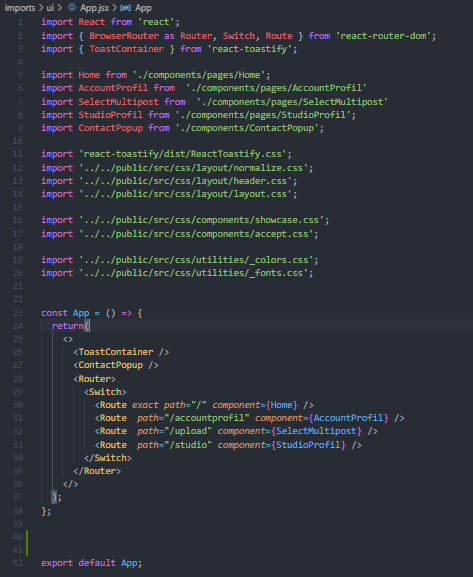
#### React-router-dom

React Router est une extension à React qui permet de gérer les routes d'une application côté client. Il permet de synchroniser (d'associer) des composants graphiques React à des urls. React Router tire profit de l'interface Windows. history pour manipuler les URL et les associer à des composants React.

Commande d’installation dans PowerShell a la racine du dossier :

$ npm install --save react-router-dom

Exemple router-dom :

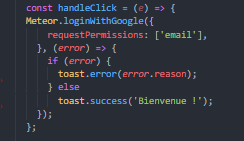


#### React-toastify

Les toasts sont des notifications légères conçues pour imiter les notifications push qui ont été popularisées par les systèmes d'exploitation mobiles et de bureau.

Commande d’installation dans PowerShell a la racine du dossier:

$ npm install --save react-toastify

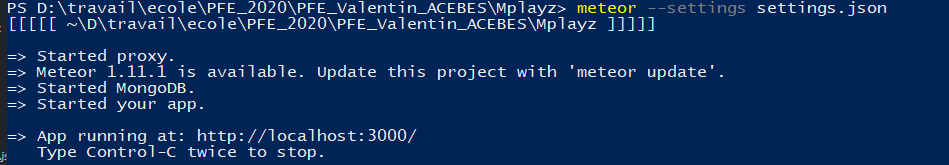


# Accounts management

### Mise en place de la connexion via Account-Google

Google dispose d'une interface de programme d'application ("API") intégrée que l’on peut utiliser pour permettre à notre application Web d'accéder à un service Google protégé par le compte Google d'un utilisateur avec l'autorisation du propriétaire du compte Google.

#### Utilisation du settings.json



#### Settings de l’API Google

{

  "private": {

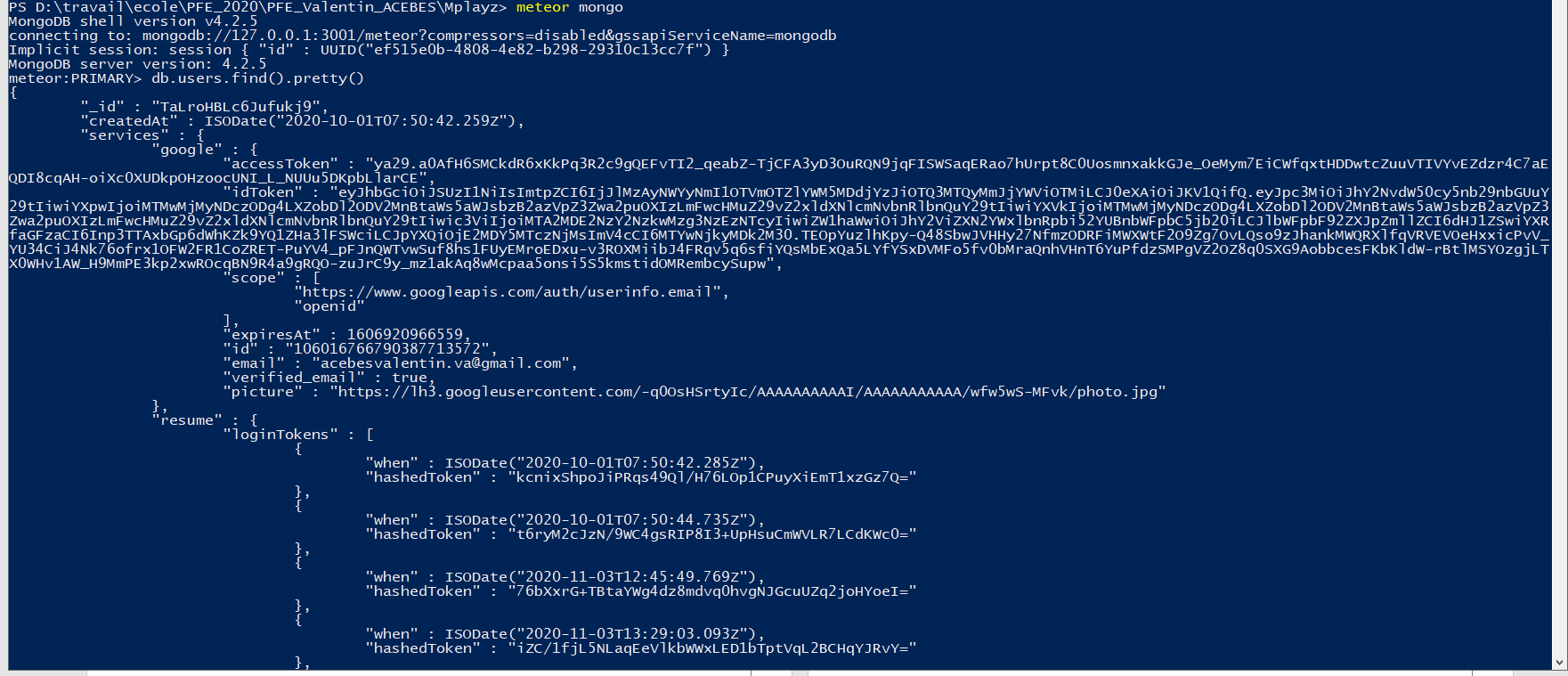
    "clientId": "130232473888-vhl9v85v2pmik9iblo0vk5igvpkjn9r3.apps.googleusercontent.com",

    "secret": "-uLIiCveZYY\_7rVc8x1uUV2l"

  }

}

### 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 données 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

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

Nous retrouverons :

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

## Fichier servis

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

Nous retrouverons :

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

## List de contrôle de sécurité

* S’assurer que l’application ne dispose pas des packages ‘insecure’ ou ‘autopublish’
* Valider tous les arguments de méthode et de publication.
* Utilisons des méthodes au lieu d’insérer/mettre à jour/supprimer coté client.
* Utiliser les sélecteurs spécifiques et champs de filtre dans les publications
* D’assurer que les clés API et les mots de passe secrets ne figurent pas dans le code source
* Sécuriser les données par l’interface utilisateur.
* Ne pas faire confiance aux ID utilisateurs transmis par le client. Utiliser this.userId à l’intérieur des méthodes et des publications.

# Contraintes techniques

La mise en place de l’UPLOAD vers les différents prend du temps, car chaque plateforme a son propre langage de développement. La prospection des API est longue et fastidieuse.

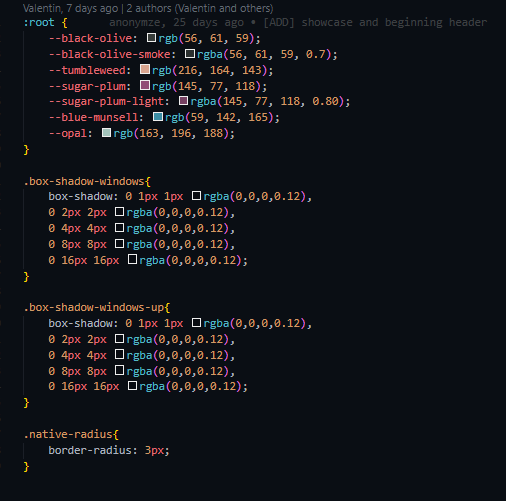
Une gestion de format de vidéo devra être proposée, car sur Twitter, les durées de vidéos sont plus courtes que sur Twitch. La plateforme Twitch a atteint le record de 200 h de flux non-stop lors d’un live sur son application.  
Les limitations des plateformes, car les démarches juridiques pour les API sont longues et des dossiers doivent être montés pour les structures de type Facebook afin de bien respecter les modalités de publication. Car lui-même surveille beaucoup les données entrantes.

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

# XVI. CSS

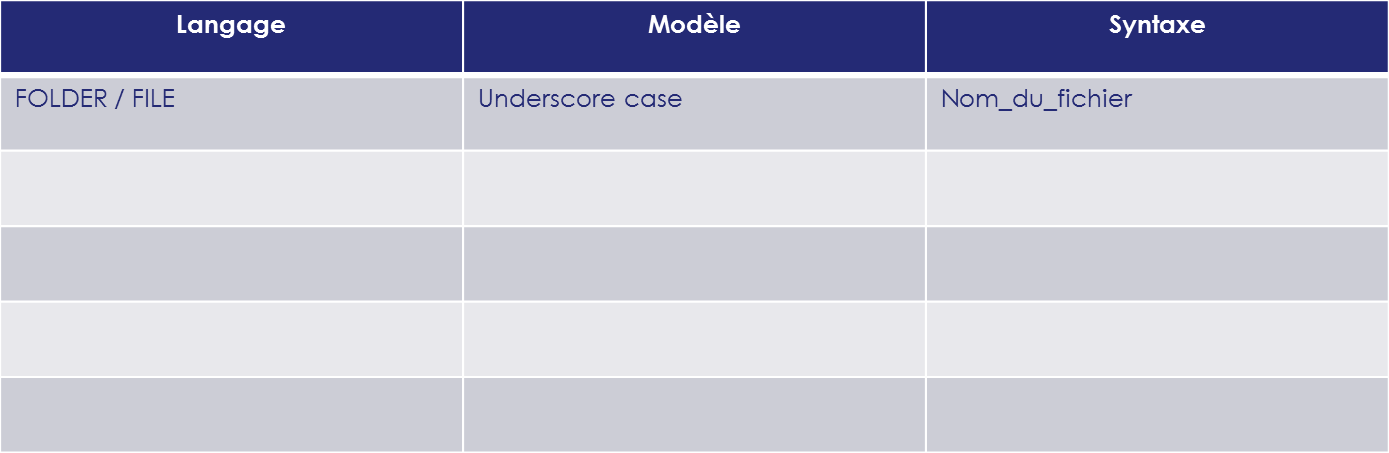
## Variable

Les propriétés personnalisées CSS (custom prospérités 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.

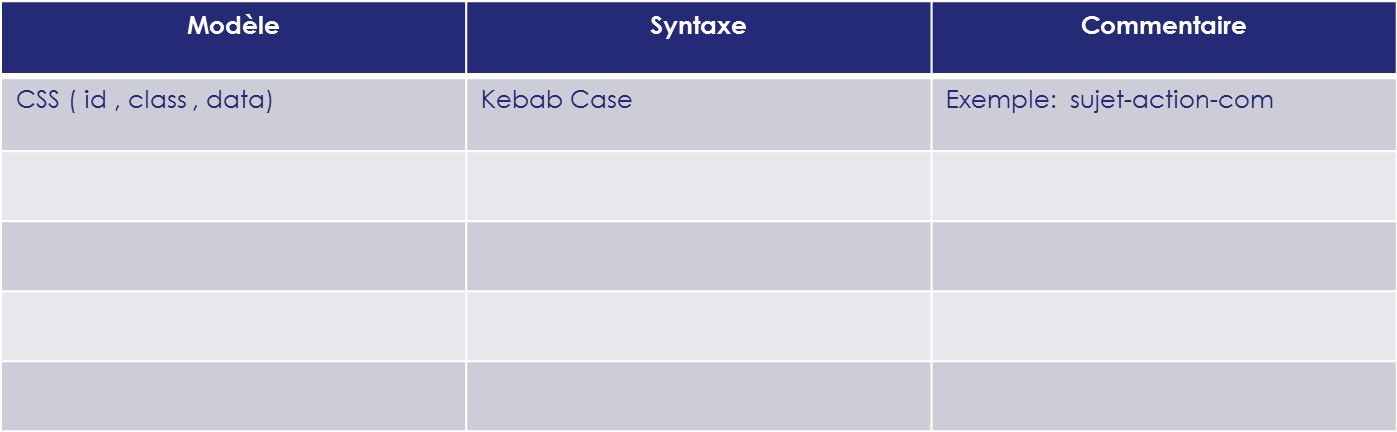


## Normalisation Naming

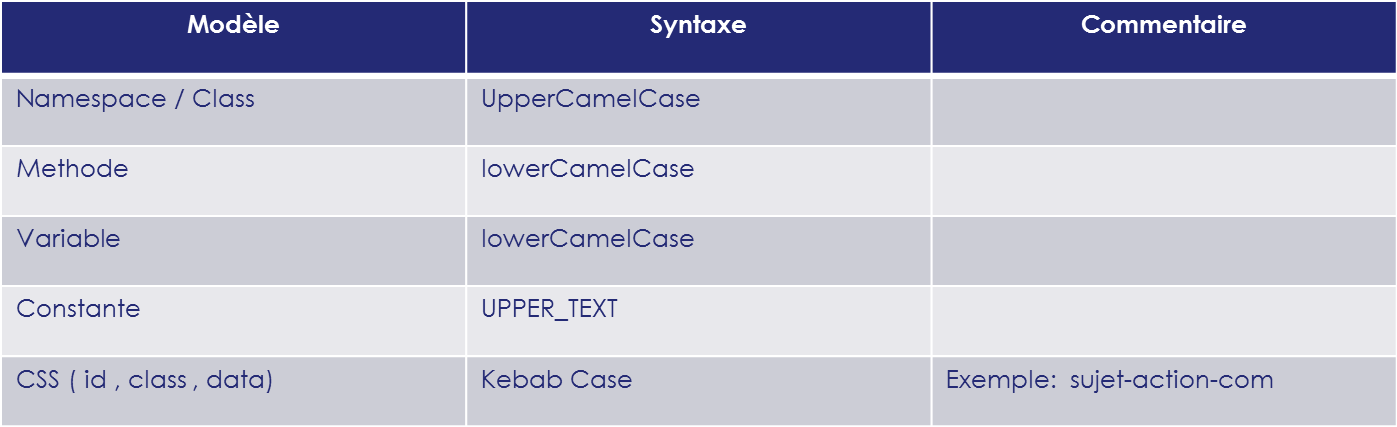
#### Syntaxes globales



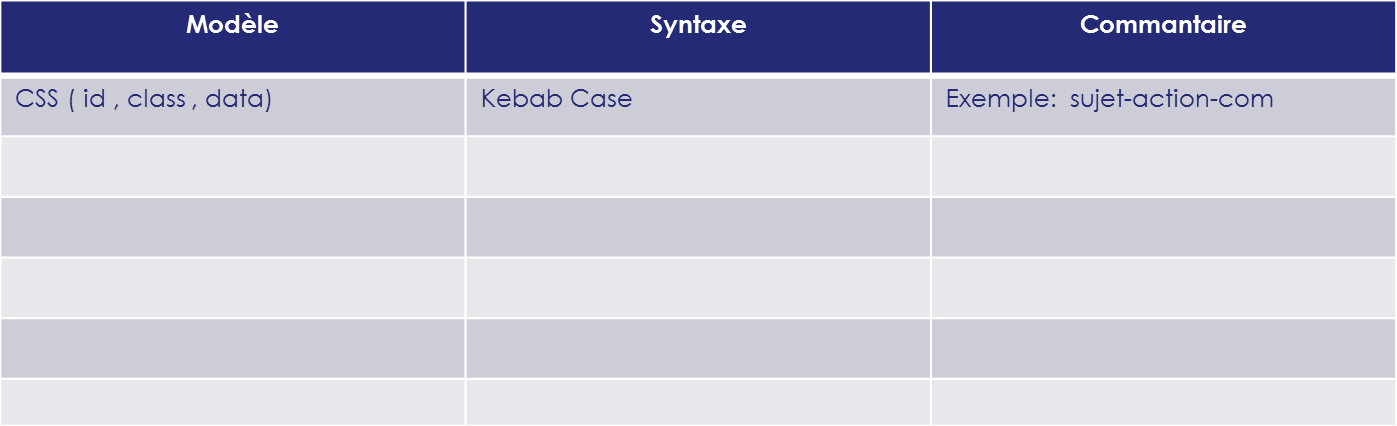
#### Syntaxes HTML



#### Syntaxes JS



#### Syntaxes CSS



## Structure CSS

Les fichiers CSS sont spécifiques 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 reconnaît à leurs underscord devant l’intitulé du fichier.

# Projection du site

## Description des principales fonctionnalités dans l’évolution

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

La mise ne place d’une connexion autre que Gmail avec plateforme tels que Facebook ou tweeter seraient l’idéal pour toucher le plus d’utilisateurs dans la simplicité de créations de comptes et de connexions aux sites.

### Plus de plateformes de publications

Des plateformes tels que Facebook ou snap sont difficiles d’accès, car les restrictions de publications sont juridiquement complexes, mais seraient un atout dans le module vidéo short clip qui sera en futur développement.

### Gestionnaire de publication

### La gestion de publications pourra permettre de supprimer/modifier une vidéo directement sur le site qui lui-même.

### Agenda de publication différé

Permettre aux utilisateurs de poster des vidéos préchargées sur un planning défini sur le site.

### Éditeur de vidéo et outils digital

## Modération d’accès

#### Free accès

La mise place d’un accès free pourrait permettre un nombre de postes et un accès aux outils limités.

#### Forfait Premium

Le forfait premium est à définir, mais il comprend dedans un nombre de postes illimité sur toutes les plateformes disponibles.

Un accès aux nouveaux outils de gestion de poste et de management.

# Conclusion

Le projet MplayZ est la première version de l’application, on peut la considérer comme une pré-alpha. Il permet lui-même de tester toutes les possibilités et réalisations pour les versions futures. J’ai développé tous les points dans leur généralité pour ensuite travailler sur des points précis afin de solidifier les différents domaines que présente ce dossier. Un projet de cette envergure nécessite un développement plus important ainsi qu'une gestion de projet conséquente. J'ai énormément appris sur différents domaines de dev, comme les librairies et frameworks React, Meteor, Node.js mais aussi de mettre en place un projet le plus rapidement possible avec les bons outils. Ces outils m’ont permis de répartir les charges de travail, et grâce à eux, j'ai eu un meilleur management et une meilleure rapidité d'exécution.

Mon souhait pour le début de mon projet a été d'utiliser la plus simple des plateformes et la plus représentative, mais par la suite, j'agrémenterais de plateformes et de fonctionnalités.

J'ai comme ambition pour la suite de créer un outil de management de médias pour le monde professionnel et public du jeu vidéo ou des activités médiatiques permettront aux différents acteurs de pouvoir organiser leurs plannings de publications ainsi que différer leurs vidéos.

# Remerciements

Tout d'abord, je tiens à remercier mes professeurs et en particulier Olivier MANGIN qui a su m'épauler et me guider durant cette formation,

Je remercie mes camarades Yann METIER et Dorian VERDON qui ont su m'épauler tout au long de ce projet.

Je remercie également Manu GUCEMAS qui m’a permis d'intégrer THALES et de ce fait de découvrir le monde de l'aéronautique.

Pour finir, merci à ma famille de m'avoir soutenu tout au long de mon apprentissage. Je tiens à souligner que cette période a été très difficile due au confinement et à la pandémie.

# Annexe