## Introduction

### Contexte et objectif

Dans le cadre du projet « toast master video uploader », l’utilisateur doit pouvoir interagir rapidement et efficacement avec l’outil. Pour permettre cette possibilité, l’interface utilisateur doit être :

* Rapide et donc léger
* Concis en information
* Accessible
* Structuré

L’objectif de la plateforme sera donc d’établir un outil web monopage et capable de fournir des fonctionnalités attendus décris dans le cahier des charges.

### Référence

En regard au document technique :

* Le cahier des charges : « BuuMySoft\_CDC\_TM »
* Le document général de conception : « TM\_DOC\_BS\_Conception générale »

## Description de la solution

### Prérequis et attente

Pour mettre en place l’aspect front en respectant les détails nommés dans le cahier des charges, nous allons énoncés les besoins détaillés et les mettre en évidence pour en retirer tous les points qu’il pourra susciter.

Il est nécessaire de devoir s’authentifier pour avoir accès à l’outil car celui-ci sollicite l’intégration de contenu sur une plateforme privée et gérée.  
L’interface doit initialiser du contenu, qui doit donc être intégré par l’utilisateur et sous plusieurs format via une formulaire.  
La gestion de la donnée doit instinctif et informationnel, et donc doit disposer de manière rapide de modifier, ré-arranger le contenu et doit pouvoir indiquer les erreurs.

Et enfin, la validation de l’export doit pouvoir indiquer si l’export s’est bien fait et si non, pour qu’elle raison.

### Pages et fonctionnalités

#### Authentification

##### Fonctionnement :

L’authentification restera une instance basique. Une mot de passe sera initialisé dans le code en tant que constante, mais bien sûr, son contenu sera haché.  
L’instance d’authentification se présentera comme ci-dessous :

[IMAGE]

Le seul champ présent est un mot de passe nécessaire pour accéder aux services de l’outil, il sera accompagné du bouton de soumission du formulaire.  
Le champ ne se mettra en rouge uniquement si le mot de passe entré est incorrect.  
Aucune autre information ne sera disponible, de sorte à ne pas donner d’indication à l’utilisateur n’ayant pas le droit d’y accéder.

##### Architecture composant :

Prérequis :

* Entité :
  + Message
* View :
  + Page de connexion
* Service :
  + AuthService
* Component :

##### Code source :

#### Import des fichiers

##### Fonctionnement :

L’import de fichier est utile car il sert à l’utilisateur pour sélectionner les fichiers qu’il veut charger dans la plateforme YouTube et charger tout autant les fichiers servant à la complétion des informations des vidéos :

[IMAGE]

Deux champs sont intégrés.  
Le premier champs a pour rôle d’importer des fichiers vidéos. Il peut en import un seul, comme il peut importer le dossier entier.  
Ce champ apparait en rouge si on vide son contenu en précisant un message d’erreur, et averti aussi si le format n’est pas le bon.

Le second champ est l’import du fichier associatif qui est imposé au format CSV. Elle complète les informations des vidéos comme leur titre, et sera mappé sur l’heure et la date défini sur l’élément.

Le bouton validera les sélections.

##### Architecture composant :

Prérequis :

* Entité :
  + Video
  + Message
* View :
  + Page des imports
* Service :
  + MessageService
  + VideoService
* Component :

##### Code source :

##### Point ouvert:

* Quel format de fichier sera toléré ?
* Y’aura-t-il une tolérence sur d’autre format de vidéo ?

#### Gestion des vidéos

##### Fonctionnement :

Une fois l’import opéré avec succès, une instance de gestion du contenu pré-importé s’affiche.

Elle liste les vidéos importés et y joint une seconde liste correspondant au information complémentaire des vidéos.   
Sur le côté droit, on y aperçoit un écran des messages d’erreur généré. Elle prévient des erreurs sur les correspondances des vidéos:

[IMAGE]

**Modification d’un élément**Dans le cas de vidéos manquantes ou en surplus, un simple message d’avertissement apparait.  
Ainsi pour lire une jointure entre deux éléments, il suffit de prêter attention sur une vidéo et l’élément se trouver au même niveau sur la liste de droite sera les informations dîtes « jointes » à la vidéo.  
Pour modifier les informations il suffit de cliquer sur l’icône du crayon, et une fois fait, les champs apparait avec les informations prédéfinis. Une fois les changements apposés, le bouton de validation permet de figer les champs.

[IMAGE]

**Modification l’emplacement d’un élément**  
S’il s’agit de modifier l’emplacement des éléments dans le liste, il suffit de placer le curseur sur l’élément de la liste, cliquer et déplacer la souris en maintenant le clique.

[IMAGE]

**Validation de la gestion** La validation des données ne peut s’opérer uniquement que si tous les messages sont traités. Une fois fait, les données sont envoyées via service de vidéo pour les traiter et les envoyés à l’interface API de YouTube.

**Gestion des messages**Les messages sont juste de simple indication des conditions nécessaire à la validation des éléments.

##### Architecture composant :

Prérequis :

* Entité :
  + Vidéo
  + Message
  + Information
* View :
  + Page de gestion
* Service :
  + MessageService
  + VideoService
* Component :
  + Video-list
  + Information-list

##### Code source :

##### Point ouvert: