# Développement Web

## Téléversement d'une image associée à un enregistrement

## Table des matières



Guy Verghote

# Téléversement d'une image associée à un enregistrement

## 1. Présentation de l'activité

Dans le cadre de l'étude de la mise en place d'un téléversement de fichier, vous êtes chargé de mettre en place un module web permettant de gérer les clubs.

Les données sont conservées dans la table club(id, nom, logo)

Le champ logo contient l'url du fichier contenant le logo du club.
**Le logo n'étant pas obligatoire**, le champ fichier accepte la valeur 'null'.

Nous accepterons les fichiers 'jpg' et 'png'
Le fichier téléversé sera renommé sans accent et avec un suffixe si ce nom existe déjà.
Le fichier ne sera pas ici redimensionné et la largeur ne devra pas dépassée 350 px.

Tous les logos sont conservés dans le répertoire data/club.

Les paramètres de configuration sont définis dans la classe métier Club

Le remplacement du logo doit être possible.

## 2. La classe métier Club

Elle dérive de la classe Table afin de pouvoir utiliser les méthodes standards assurant les opérations de gestion sur une table MySQL.

Les paramètres de configuration du fichier téléchargé (le logo) sont conservés dans un tableau CONFIG au sein de la classe, et peuvent être récupérés via la méthode getConfig().

```
private const CONFIG = [
        'repertoire' => '/data/club',
        'extensions' => ["jpg", "png"],
        'types' => ["image/pjpeg", "image/jpeg", "x-png", "image/png"],
        'maxSize' => 150 * 1024,
        'require' => false,
        'rename' => true,
        'sansAccent' => true,
        'redimensionner' => false,
        'height' => 0,
        'width' => 350,
        'accept' => '.jpg, .png',
        'label' => '(150 Ko max, jpg ou png)',
];
```

```
public static function getConfig(): array  {
        return self::CONFIG;
}
```

Le constructeur de la classe décrit chaque champ composant la table Club à l'aide d'objets dérivés de la classe Input afin d'encapsuler dans chaque champ les règles de validité.

```
public function __construct() {
        parent::__construct('club');

        // Colonne id : clé primaire (6 chiffres)
        $input = new inputText();
        $input->Require = true;
        $input->Pattern = "^[0-9]{6}$";
        $input->MaxLength = 6;
        $input->MinLength = 6;
        $this->columns['id'] = $input;

        // Colonne nom
        $input = new inputText();
        $input->Require = true;
        $input->Casse = 'U';
        $input->SupprimerAccent = true;
        $input->SupprimerEspaceSuperflu = true;
        $input->Pattern = "^[A-Z0-9]+([ .\-]?[A-Z0-9]+)*$";
        $input->MaxLength = 70;
        $this->columns['nom'] = $input;

        // Colonne logo
        $input = new InputText();
        $input->Require = false;
        $this->columns['logo'] = $input;

        // Définition des colonnes modifiables en mode colonne
        $this->listOfColumns->Values = ['nom'];
    }
```

Le constructeur de la classe Club décrit la structure d'un enregistrement de la table club

La colonne id est la clé primaire de la table club mais sa particularité est de ne pas être un champ auto-incrémenté, il faut donc la définir car sa valeur sera saisie par l'utilisateur.
La valeur est obligatoire
Elle doit être composée de 6 chiffres exactement
L'objet est stocké dans le tableau associatif contenu toutes les colonnes dont la saisie est nécessaire

La colonne 'nom' est représenté par un objet intputText
Sa valeur est obligatoire, elle sera mise automatiquement en majuscule, les accents seront supprimés
Les espaces superflus aussi
Sa value doit commencer par une lettre ou un chiffre suivi éventuellement d'un '.' ou d'un '–' lui-même suivi par au moins une lettre ou un chiffre
Elle ne doit pas dépassée 70 caractères

La colonne nom pourra être modifiée en mode colonne

La colonne 'logo' est représentée par un objet InputText.
Sa valeur est facultative (valeur 'null' acceptée)

Comme pour la gestion des documents nous retrouvons les 4 méthodes suivantes :

La méthode **getAll()** retourne les enregistrements en ajoutant une colonne 'present' avec la valeur 1 si le fichier défini dans le champ fichier est bien présent dans le répertoire /data/gestiondocument
La méthode **getById($id)** retourne les informations sur un club
La méthode **supprimerFichier($id)** permet de supprimer un fichier (logo) dans le répertoire de stockage
La méthode **supprimer($id)** permet de supprimer un enregistrement dans la table

Une nouvelle méthode est nécessaire afin de pouvoir remplacer le logo car cette fois-ci le remplacement engendre un changement dans la table

```
public static function majLogo(string $id, string $logo): void   {
        $sql = "update club set logo = :logo where id = :id;";
        $db = Database::getInstance();
        $curseur = $db->prepare($sql);
        $curseur->bindValue('id', $id);
        $curseur->bindValue('logo', $logo);
        $curseur->execute();
}
```

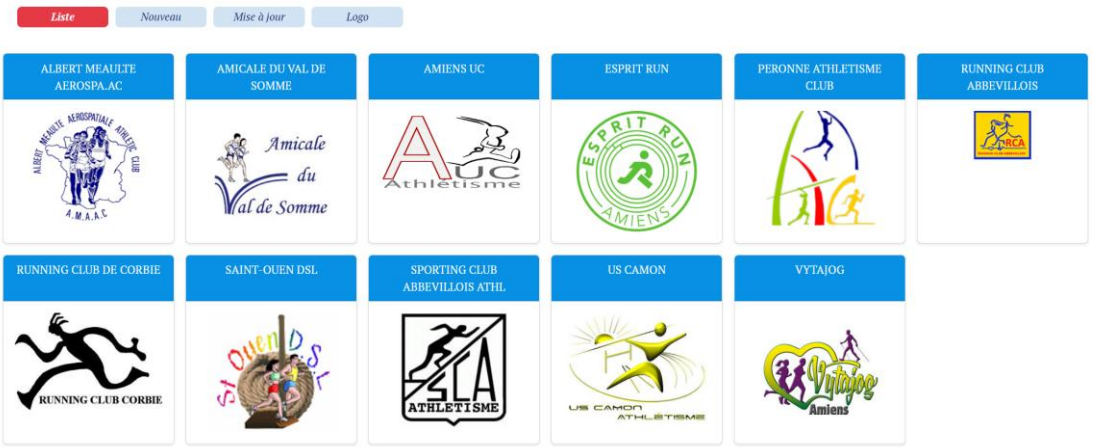
### 3. Les contrôles mis en place au niveau de la base de données

Le déclencheur 'avantAjoutClub' permet de vérifier les contrôles portant sur la mise en forme des données mais surtout il met en place des contrôles supplémentaires vérifiant la validité des données par rapport aux contraintes portant sur la table : unicité de la clé primaire, du nom

```
create trigger avantAjoutClub before insert on club
for each row
begin
   -- Vérification sur l'id
   if new.id REGEXP '^[0-8]{3}[0-9]{3}$' = 0 THEN
      signal sqlstate '45000' set message_text =
            'Le numéro du club doit être un nombre de 6 chiffres commençant par 080';
   end if;
   if exists(select 1 from club where id = new.id) then
      signal sqlstate '45000' set message_text = '#Ce numéro est déjà attribué';
   end if;
   -- Mise en forme et vérification sur le nom
   set new.nom = ucase(new.nom);
   if char_length(new.nom) not between 3 and 60 then
      signal sqlstate '45000' set message_text = '#Le nom doit comprendre entre 3 et 60 caractères';
   end if;
   if new.nom not regexp '^[A-Za-z]+([ "\-.]?[A-Za-z])*$' THEN
      signal sqlstate '45000' set message_text = '#Le format du nom est invalide.';
   end if;
   if exists(select 1 from club where nom = new.nom) then
      signal sqlstate '45000' set message_text = '#Ce nom est déjà utilisé';
   end if;
end;
```

*Le déclencheur 'avantModificationClub' permet de faire les mêmes contrôles avant la modification d'un club.*
*La différence résida dans l'ajout d'un test comparant la nouvelle valeur à l'ancienne afin de déclencher le contrôle.*

## 4. La page d'accueil du module




Ce point de terminaison permet à partir d'un menu horizontal :

- de consulter les clubs dans une mise en forme de type 'carte',
- d'ajouter un nouveau club
- de modifier en mode colonne le nom du club ou de supprimer un club,
- de remplacer le logo d'un club,

Le menu horizontal est géré à l'aide du composant 'composant/menuhorizontal'
Le paramétrage du menu s'effectue par la lecture du fichier menuhorizontal.json présent dans le sous répertoire .config du module comme pour le module précédent.

## 5. L'interface permettant la consultation

### 5.1. Le script liste/index.php

Il transmet côté client, les enregistrements de la table Club et le répertoire de stockage, seul paramètre nécessaire pour pouvoir alimenter côté client l'attribut 'src' des balises img générées dynamiquement.

```
// récupération des clubs
$lesClubs = json_encode(Club::getAll(), JSON_UNESCAPED_UNICODE |
JSON_UNESCAPED_SLASHES);

// récupération du répertoire contenant les logos des clubs
$repertoire = json_encode(Club::getConfig()['repertoire'],
JSON_UNESCAPED_UNICODE | JSON_UNESCAPED_SLASHES);

$head = <<<HTML
   <script>
      const lesClubs = $lesClubs ;
      const repertoire = $repertoire;
   </script>
HTML;
```

1. **Compléter le script index.php.**

### 5.1. Le fichier index.js

Il prend en charge l'affichage des clubs avec une mise en forme de type 'carte'

Les cartes sont générées par la fonction creerCarte :

```
function creerCarte(element) {
    // Création de la carte avec classe spécifique
    const carte = document.createElement('div');
    carte.classList.add('card', 'carte-club', 'shadow-sm');

    // En-tête
    const entete = document.createElement('div');
    entete.classList.add('card-header', 'text-center');
    entete.style.height = '80px';
    entete.style.backgroundColor =
getComputedStyle(document.documentElement).getPropertyValue('--background-color-
header').trim();
    entete.style.color = getComputedStyle(document.documentElement).getPropertyValue('--text-
color-header').trim();
    entete.innerText = element.nom;

// Création du corps de la carte contenant le logo
    const corps = document.createElement('div');
    corps.classList.add("card-body");
    corps.style.height = '250px'; // Hauteur fixe pour uniformiser les cartes

    if (element.present) {
        const img = document.createElement('img');
        img.src = repertoire + '/' + element.fichier;
        img.alt = `${element.nom} logo`;
        // Ajoute les styles pour que l'image reste dans le conteneur
        img.style.maxHeight = '100%';
        img.style.maxWidth = '100%';
        img.style.objectFit = 'contain';
        img.style.display = 'block';
        img.style.margin = '0 auto'; // centrer horizontalement
        corps.appendChild(img);
    }
    carte.appendChild(entete);
    carte.appendChild(corps);
    return carte;
}
```

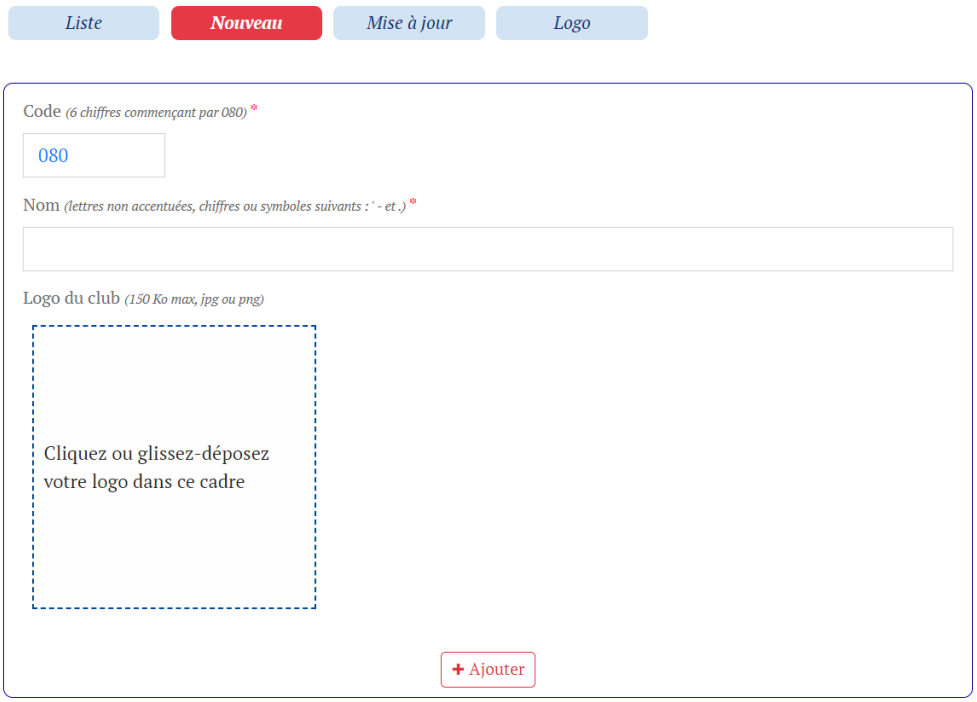
Le programme principal parcourt les clubs afin de générer chaque carte.
Les cartes sont placées dans un conteneur utilisant le mode 'flex' afin d'obtenir un affichage responsif

```
// Création d'un conteneur flex
let conteneur = document.createElement('div');
conteneur.style.display = 'flex';
conteneur.style.flexWrap = 'wrap';
conteneur.style.gap = '1rem'; // Espacement entre les cartes
conteneur.style.justifyContent = 'flex-start';
```

```
// Création des cartes
for (const element of lesClubs) {
    const carte = creerCarte(element);
    // Limite de taille et comportement responsif
    carte.style.flex = '0 1 300px'; // Ne grandit pas, peut rétrécir, base 300px
    carte.style.maxWidth = '100%';
    conteneur.appendChild(carte);
}
lesCartes.appendChild(conteneur);
```

**2. Compléter le script index.js et réaliser les tests fonctionnels de cette fonctionnalité.**

### 6. L'interface permettant l'ajout d'un club



### 6.1. Le script ajout/ajax/ajouter.php

Le logo étant facultatif, le contrôle est conditionné à la transmission d'un fichier de même que la copie dans le répertoire de stockage

```
<?php
// Si un fichier est téléversé, il faut le controler
if (isset($_FILES['fichier'])) {
    // instanciation et paramétrage d'un objet InputFile
    $file = new InputFile($_FILES['fichier'], Club::getConfig());
    // vérifie la validité du fichier
    if (!$file->checkValidity()) {
        Erreur::envoyerReponse($file->getValidationMessage(), 'global');
    }
}
// création d'un objet Club pour réaliser les contrôles sur les données
$Club = new Club();

// Les données ont-elles été transmises ?
if (!$Club->donneesTransmises()) {
    Erreur::envoyerReponse("Toutes les données attendues ne sont pas transmises", 'global');
}

// Toutes les données sont-elles valides ?
if (!$Club->checkAll()) {
    Erreur::envoyerReponse("Certaines données transmises ne sont pas valides", 'global');
}

// Alimentation éventuelle de la colonne 'logo'  : sa valeur est stockée dans la propriété Value de l'objet $file
if (isset($file)) {
    $Club->setValue('logo', $file->Value);
}
// Ajout dans la table Club
$Club->insert();

// Récupération de l'identifiant du Club ajouté
$id =  $Club->getLastInsertId();

// copie éventuelle du fichier dans le répertoire de stockage
if (isset($file)) {
    $ok = $file->copy();
    // en cas d'échec (peu probable) il faut supprimer l'enregistrement créé afin de conserver une cohérence
    if (!$ok) {
        $Club->delete($id);
        Erreur::envoyerReponse("L'ajout a échoué car le logo n'a pas pu être téléversé", 'global');
    }
}
// Tout est OK
$reponse = ['success' => $id];
echo json_encode($reponse, JSON_UNESCAPED_UNICODE);
```

### 6.2. Le script ajout/index.php

Il récupère les paramètres du téléversement en appelant la méthode getConfig de la classe club afin de les envoyer vers le client.

```
// récupération des paramètres de configuration pour le logo des clubs
$lesParametres = json_encode(Club::getConfig());

$head = <<<HTML
   <script>
      const lesParametres = $lesParametres;
   </script>
HTML;
```

**3. Compléter le script index.php.**

### 6.3. Le script ajout/index.js

Le contrôle du fichier téléversé est pris en charge par la fonction controlerFichier(file) déclenchée suite au téléversement d'un fichier image
Le téléversement est déclenché par le clic dans la zone 'cible' ou en faisant glisser un fichier dans le cadre.

```
// Déclencher le clic sur le champ de type file lors d'un clic dans la zone cible
cible.onclick = () => fichier.click();

// ajout du glisser déposer dans la zone cible
cible.ondragover = (e) => e.preventDefault();
cible.ondrop = (e) => {
   e.preventDefault();
   controlerFichier(e.dataTransfer.files[0]);
};
// Lancer la fonction controlerFichier si un fichier a été sélectionné dans l'explorateur
fichier.onchange = () => {
   if (fichier.files.length > 0) {
      controlerFichier(fichier.files[0]);
   }
};
```

Le clic sur le bouton 'Ajouter ' déclenche l'appel de la fonction ajouter() après avoir vérifié la validité des données saisies

```
btnAjouter.onclick = () => {
    effacerLesErreurs();
   if (donneesValides()) {
      ajouter();
   }
};
```

Le contrôle du fichier téléversé est pris en charge par la fonction controlerFichier(file)

```
function controlerFichier(file) {
   // Efface les erreurs précédentes
   effacerLesErreurs();
   // Vérification de taille et d'extension
   if (!fichierValide(file, lesParametres)) {
      return;
   }
   // si le redimensionnement n'est pas demandé, on vérifie les dimensions
   if (!lesParametres.redimensionner) {
      verifierDimensionsImage(file, lesParametres,
         (file, img) => {
            nomFichier.innerText = file.name;
            leFichier = file;
            cible.innerHTML = "";
            cible.appendChild(img);
         }
      );
   }
}
```

*La fonction verifierDimensionsImage() du même module vérifie les dimensions définies dans les paramètres de configuration. Cela nécessite le chargement de l'image en mémoire ce qui rend la fonction asynchrone. Il faut attendre le chargement de l'image pour pouvoir vérifier ses dimensions.*

Ces fonctions possède 3 arguments :

- L'objet file
- Les paramètres à respecter
- La fonction de rappel à exécuter en cas de succès

La fonction de rappel reçoit implicitement 2 paramètres :

- Le fichier téléverse (l'objet file)
- L'objet Image qui a été chargé en mémoire par la fonction verifierDimensionsImage

L'objet file est conservé dans la variable globale leFichier et l'objet Image est encapsulé dans la div 'cible'

La fonction **ajouter()** doit transmettre l'identifiant du club, le nom du club et le champ de type file Pour cela il faut obligatoirement utiliser un objet formData

```
function ajouter() {
   effacerLesErreurs();
   const formData= new FormData();
   formData.append('id', id.value);
   formData.append('nom', nom.value);
   // la photo n'est pas obligatoire
   if (leFichier !== null) {
      formData.append('fichier', leFichier);
   }
   appelAjax({
      url: 'ajax/ajouter.php',
      data: formData,
      success: () => retournerVersApresConfirmation("Le club a été ajouté", '../liste')
   });
}
```

**4. Compléter le script ajout.js et réaliser tous les tests fonctionnels.**

## 7. L'interface permettant la modification du nom ou la suppression d'un club


Nous avons opté pour une mise en forme de type tableau avec la possibilité de modifier directement la colonne nom, la table club comportant peu d'enregistrements et la modification du numéro n'est pas autorisé (voir déclencheur)
La première colonne comporte une icône permettant de demander la suppression du club.

### 7.1. Le script maj/ajax/supprimer.php

Comme pour l'ajout, il n'est pas possible d'utiliser le script standard /ajax/supprimer.php car il faut supprimer un enregistrement et aussi supprimer éventuellement le fichier image associé.

```
<?php
// Contrôle de l'existence du paramètre attendu : id
if (!isset($_POST['id'])) {
    Erreur::envoyerReponse("Paramètre manquant", 'global');
}
// récupération du paramètre attendu
$id = $_POST['id'];

// vérification de l'existence du club
$ligne = Club::getById($id);
if (!$ligne) {
    Erreur::envoyerReponse("Ce club n'existe pas", 'global');
}
// suppression de l'enregistrement en base de données
Club::supprimer($id);

// suppression du fichier image associé
if (!empty($ligne['logo'])) {
    Club::supprimerFichier($ligne['logo']);
}

$reponse = ['success' => "Le Club a été supprimé"];
echo json_encode($reponse, JSON_UNESCAPED_UNICODE);
```

**5. Compléter le script supprimer.php et réaliser tous les tests fonctionnels.**

### 7.2. Le script maj/index.php

Il transmet côté client les enregistrements de la table club en appelant la méthode getAll() de la classe Club

**6. Compléter le script index.php**

### 7.3. Le script maj/index.js

La fonction **supprimer()** déclenche l'appel ajax vers le script ajax/supprimer.php

```
function supprimer(id) {
   appelAjax({
      url: 'ajax/supprimer.php',
      data: {id: id},
      success: () => document.getElementById(id.toString())?.remove()
   });
}
```

La fonction **modifierNom()** déclenche l'appel Ajax vers le script standard /ajax/modifiercolonne.php

```
function modifierNom(input, id) {
   appelAjax({
      url: '/ajax/modifiercolonne.php',
      data: { table: 'club',  colonne: 'nom',  valeur: input.value,  id: id },
      success: () => input.style.color = 'green'
   });
}
```

Nous avons maintenant besoin de fonctions permettant de générer les cellules du tableau affiché

La fonction **creerColonneAction()** retourne une balise <td> contenant le bouton d'action 'supprimer'

```
function creerColonneAction(id) {
   const actionSupprimer = () => confirmer(() => supprimer(id));
   const btnSupprimer = creerBoutonSuppression(actionSupprimer);
   btnSupprimer.setAttribute('aria-label', 'Supprimer le club');
   const tdAction = getTd('');
   tdAction.appendChild(btnSupprimer);
   return tdAction;
}
```

Le bouton supprimer lancera la fonction supprimer() après confirmation

La fonction **creerColonneNom()** retourne une balise <td> contenant la balise input permettant de modifier la valeur du nom du document. L'événement change, associé à la balise input permet de déclencher la fonction modifierNom()

```
function creerColonneNom(id, nom) {
   const inputNom = document.createElement('input');
   inputNom.type = 'text';
   inputNom.value = nom;
   inputNom.required = true;
   inputNom.maxLength = 70;
   inputNom.pattern = "^[A-Za-z]+([ '\\-.]?[A-Za-z0-9])*$";
   inputNom.onchange = function () {
      this.value = enleverAccent(this.value).toUpperCase();
      if (this.checkValidity()) {
         modifierNom(this, id);
      } else {
         this.style.color = 'red';
         this.reportValidity();
      }
   };
   const tdNom = getTd('');
   tdNom.appendChild(inputNom);
   return tdNom;
}
```

La fonction **creerLigneClub()** retourne une balise <tr> contenant les trois balises <td> générées par les fonctions creerColonneAction(), getTd() et creerColonneTitre

```
function creerLigneClub(club) {
   const {id, nom} = club;

   // Création de la ligne (nécessaire pour la référence circulaire)
   const tr = getTr([]);
   tr.id = id;

    // Création des colonnes
   const tdAction = creerColonneAction(id);
   const tdId = getTd(id);
   const tdNom = creerColonneNom(id, nom);

   // Ajout des cellules à la ligne
   tr.appendChild(tdAction);
   tr.appendChild(tdId);
   tr.appendChild(tdNom);

   return tr;
}
```

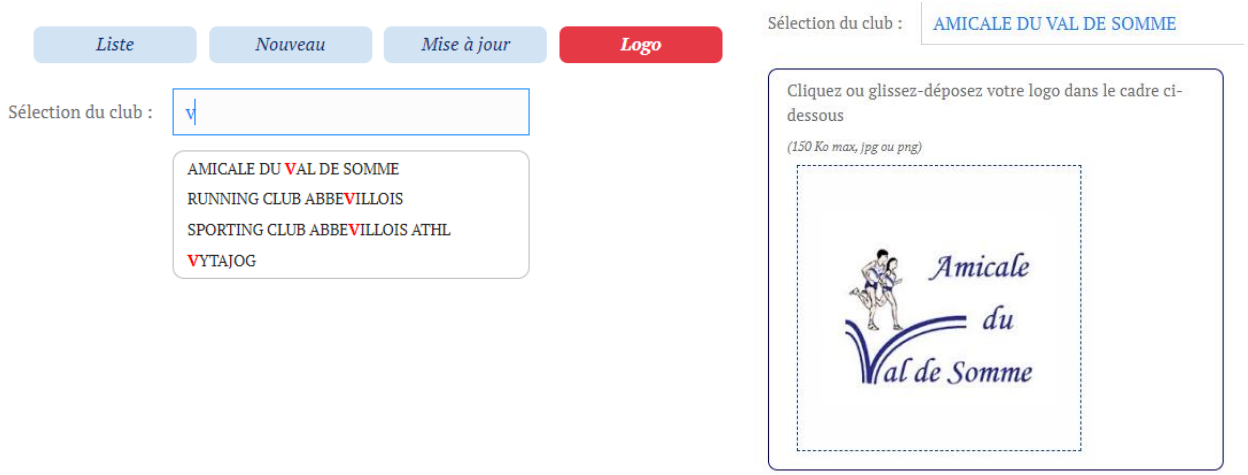
Il ne reste plus qu'à afficher le tableau

```
for (const club of lesClubs) {
   lesLignes.appendChild(creerLigneClub(club));
}
```

**7. Compléter le script index.js et réaliser tous les tests fonctionnels nécessaires**

## 8. L'interface permettant le remplacement d'un logo

Nous décidons ici de ne pas opter pour une interface en mode colonne comme dans le module précédent mais une interface avec sélection du club par une zone avec autocomplétion.




Le script logo/index.html contient un champ de type file afin de pouvoir téléverser le nouveau document.

```
<input type="file" id="fichier" accept="" style='display: none '>
```

### 8.1. Le script logo/ajax/modifier.php

```
<?php
// contrôle de la présence du fichier transmis
if (!isset($_FILES['fichier'])) {
    Erreur::envoyerReponse("Le nouveau logo du club n'est pas transmis", 'global');
}

// instanciation et paramétrage d'un objet InputFile
$file = new InputFileImg($_FILES['fichier'], Club::getConfig());

// vérifie la validité du fichier
if (!$file->checkValidity()) {
    Erreur::envoyerReponse($file->getValidationMessage(), 'global');
}

// vérification du paramètre id :
if (!isset($_POST['id']) || empty($_POST['id'])) {
    Erreur::afficherReponse("L'identifiant du club concerné n'est pas transmis", 'global');
}

// récupération de l'identifiant du club
$id = $_POST['id'];

// contrôle de la validité du paramètre
if (!preg_match('/^080[0-9]{3}$/', $id)) {
    Erreur::bloquerVisiteur();
}
```

```
// vérifier l'existence du club
$club = Club::getById($id);
if (!$club) {
   Erreur::traiterReponse("Ce club n'existe pas", 'global');
}

// copier le nouveau logo
$ok = $file->copy();
if (!$ok) {
   Erreur::envoyerReponse("Le téléversement du nouveau logo a échoué", 'global');
}

// supprimer l'ancien logo s'il existe et si son nom est différent de celui du nouveau logo
if (!empty($club['logo']) && $club['logo'] !== $file->Value) {
   Club::supprimerFichier($club['logo']);
}

// mettre à jour le champ logo de l'enregistrement
Club::majLogo($id, $file->Value);
echo json_encode(['success' => 'Le logo a été enregistré']);
```

**8. Compléter le script ajax/modifier.php.**

### 8.2. Le script logo/index.php

Il transmet côté client les clubs et les paramètres liés au téléversement et charge le composant autocomplete

```
// récupération des paramètres de configuration pour le logo des clubs
$lesParametres = json_encode(Club::getConfig());
$lesClubs = json_encode(Club::getAll());

$head = <<<HTML
   <script src="/composant/autocomplete/autocomplete.min.js"></script>
   <link rel="stylesheet" href="/composant/autocomplete/autocomplete.css">
   <script>
      const lesParametres = $lesParametres;
      const lesClubs = $lesClubs;
   </script>
HTML;
```

**9. Compléter le script index.php.**

### 8.3. Le script logo/index.js

Il est possible de sélectionner un fichier ou de le faire glisser directement dans la 

```
// Déclencher le clic sur le champ de type file lors d'un clic dans la zone cible
cible.onclick = () => fichier.click();
// ajout du glisser déposer dans la zone cible
cible.ondragover = (e) => e.preventDefault();
cible.ondrop = (e) => {
   e.preventDefault();
   controlerFichier(e.dataTransfer.files[0]);
};
// traitement du champ file associé aux modifications de photos
fichier.onchange = function () {
   if (this.files.length > 0) {
      controlerFichier(this.files[0]);
   }
};
```

sur la réception du focus sur le champ de recherche de club il faut vider le champ et masquer le formulaire

```
nomR.onfocus = () => {
   // on efface les erreurs précédentes
   effacerLesErreurs();
   // on masque le formulaire
   formulaire.style.display = 'none';
   // on vide la zone cible
   cible.innerHTML = '';
   // on vide le champ
   nomR.value = '';
};
```

Le contrôle du fichier téléversé est pris en charge par la fonction controlerFichier

```
function controlerFichier(file) {
   // Efface les erreurs précédentes
   effacerLesErreurs();
   // Vérification de taille et d'extension
   if (!fichierValide(file, lesParametres)) {
      return;
   }
   // Vérifications spécifiques pour un fichier image
   // la fonction de rappel reçoit le fichier et l'image éventuellement redimensionnée si le
redimensionnement est demandé
   verifierImage(file, lesParametres, majLogo);
}
```

La fonction verifierImage est une fonction asynchrone car elle doit attendre le chargement de l'image pour savoir s'il s'agit bien d'une image et pour vérifier, si demandé ses dimensions. Il faut donc utiliser une fonction de rappel appelée en cas de succès pour continuer le traitement

La fonction majLogo (file, img) réalise un appel Ajax du script ajax/modifier.php

```
function majLogo(file, img) {
    const formData = new FormData();
    formData.append('fichier', file);
    formData.append('id', club.id);
    appelAjax({
        url: 'ajax/modifier.php',
        data: formData,
        success: () => {
            afficherToast("Le logo a été mis à jour");
            cible.innerHTML = "";
            cible.appendChild(img);
        }
    });
}
```

**10. Compléter le script remplacement.js et réaliser tous les tests fonctionnels.**