

S3.01 - Développement d’application et Gestion de Projet

Enseignant tuteur : Nathalie VALLES-PARLANGEAU

Groupe n°1

**SAE du Troisième Semestre**

Pôle Gestion de Données

Rapport Final



Sport Track

Application Web Responsive de gestion de clubs sportifs amateurs

Matis Chabanat | Titouan Cocheril

Arthur Le Menn | Ivan Salle

- - -

TD1 | TP1

BUT Informatique - Parcours A

Semestre 3

- 2022 | 2023 -

# Sommaire

[**Sommaire**](#_o1i08bj8ximl) **2**

[**1 - Pitch de l’application**](#_two4yelj3fqa) **3**

[**2 - Problème de stockage**](#_y082tv6vpx2t) **4**

[**3 - Les entitées**](#_elf7jutdlpmg) **4**

[**4 - Schéma UML**](#_akcs0vjqhm1j) **6**

[**5 - Modèle relationnel**](#_4poks54qbztd) **7**

[**6 - Dictionnaire de données**](#_p3b4bizeb9i) **7**

[**7 - Manipulation de la Base de données**](#_lyr2f8f76049) **11**

[7.1 - Peuplement de la base de données](#_i2phglfakjxi) 11

[7.2 - Requêtes sur la base de données](#_b7x87we9cmhn) 15

[**8 - Technologie utilisées**](#_pvmid04f3tdm) **16**

[**9. Consulter Scripts et base de données**](#_br94pb3axrlf) **17**

# 1 - Pitch de l’application

Sport Track est une application web de gestion de club sportif amateur. Elle est responsive c'est-à-dire qu’elle s’adapte pour tous types d'appareils.

La cible de notre application sont les sportifs amateurs licenciés, les dirigeants et entraîneurs de clubs sportifs amateurs.

L’application se divisera en trois grandes fonctionnalités : une partie messagerie qui permettra aux acteurs de l’application de communiquer. L'entraîneur pourra communiquer avec ses joueurs et le dirigeant de son club. Mais les joueurs pourront aussi parler entre eux.

La seconde partie de l’application sera la partie calendrier : les joueurs pourront consulter leurs entraînements, matchs avec le lieu et l’horaire, l'entraîneur pourra planifier ses entraînements, le dirigeant pourra gérer l’accès à son club et son terrain/salle de sport.

La dernière partie de l’application sera une partie profil : statistiques, classement, palmarès et composition équipe mais aussi des équipes adverses.

On a décidé que l’application sera destinée pour 1 sports seulement : le football : 660 000 licenciés. En ajoutant tous les dirigeants de club et les entraîneurs, l’application serait donc destinée à un grand nombre de personnes.

On peut ajouter qu’on laissera accès à l’application qu’au sportifs licenciés, entraîneurs et dirigeants confirmés pour créer un réel écosystème de sportif.

L’application leur servira d’outil pour faciliter leur épanouissement au sein du club.



# 2 - Problème de stockage

Pour répondre au besoin de notre application, nous avons besoin de deux sources de données :

* Base de données / API de la fédération

Afin de récupérer les données des joueurs (nom, prenom, poste, licence), des clubs, des matchs, des championnats etc …

* Base de données de Sport Track

Afin de stocker les données d’inscription, les messages, et enfin les événements du calendrier.

**Attention ⚠** : Nous n’avons pas réussi à obtenir la base de données officielle de la fédération, pour remédier à ce problème nous allons créer et peupler une base fictive qui simulera divers championnats. De ce fait, toutes nos données seront dans une seule et même base de données.

# 3 - Les entitées

Notre base de données sera composée de 9 classes d’entités et de 1 classes d’association :

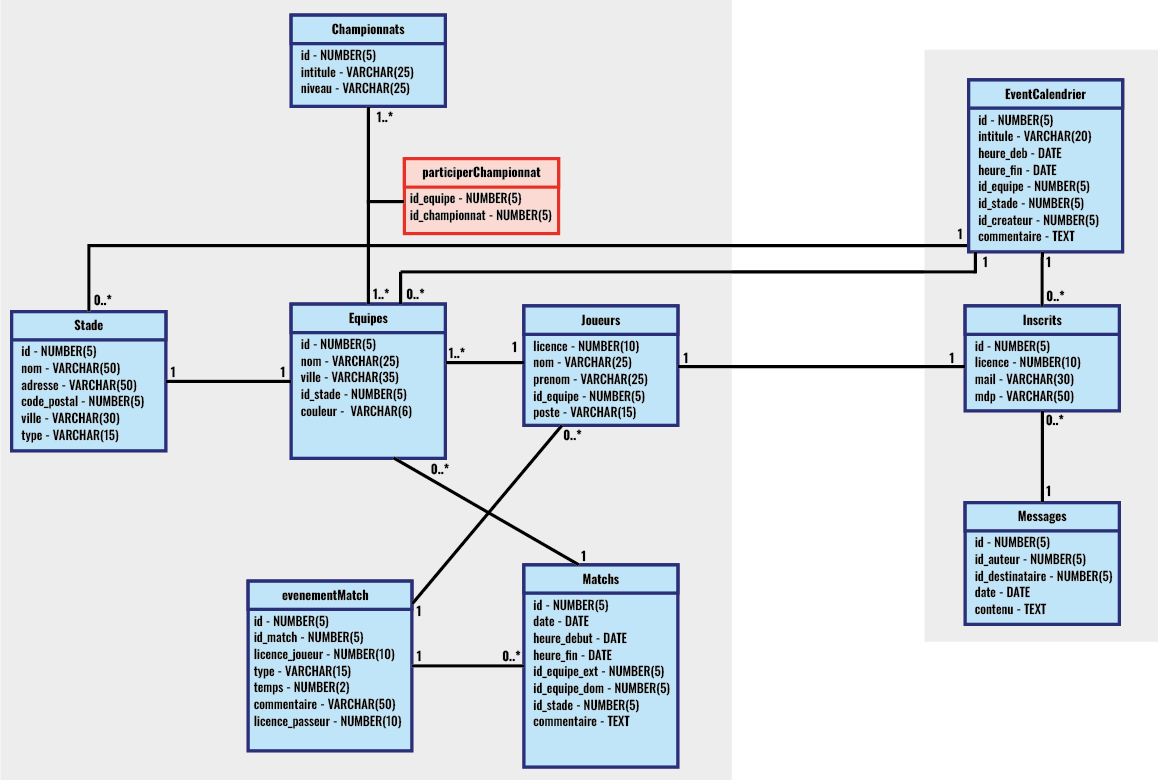
**Classes d’entités**

| **Nom** | **Description** |
| --- | --- |
| Stade | Stade de football |
| Championnat | Championnat contenant plusieurs équipes |
| Equipe | Equipe de football composée de joueurs |
| Joueur | Joueur de football |
| Match | Match de football |
| EvenementMatch | Événement se passant lors des match (But, carton etc) |
| EventCalendrier | Evénement du calendrier |
| Inscrit | Les profils inscrits |
| Message | Les messages envoyés depuis la messagerie |

**Classe d'association**

| **Nom** | **Description** |
| --- | --- |
| participerChampionnat | la participation d'une équipe à un championnat |

# 4 - Schéma UML



Commentaire :

La partie gauche représente la base de donnée que nous aurions dû obtenir avec la fédération, la partie droite, elle représente la base de donnée de notre application. Étant donné que l’on simule la base de donnée de la fédération, nous avons décidé de tout mettre dans la même base afin de nous simplifier la tâche.

# 5 - Modèle relationnel

Stade (**id**, nom, adresse, code\_postal, ville, type);

Equipe (**id**, nom, ville, **#id\_stade, couleur**);

Championnat (**id**, intitule, niveau);

ParticiperChampionnat (**#id\_equipe, #id\_championnat**);

Joueur (**licence**, nom, prenom, **#id\_equipe, poste**);

MatchTable (**id**, jouer, heure\_debut, heure\_fin, **#id\_equipe\_dom**, **#id\_equipe\_ext, #id\_stade**, journee, commentaire);

EvenementMatch (**id**, **#id\_match, #licence\_joueur**, type, temps, commentaire);

Inscrit (**id**, licence, mail, mdp);

Message (**id**, **#id\_auteur**, **#id\_destinataire**, date, contenu);

EventCalendrier (**id**, intitule, heure\_deb, heure\_fin, **#id\_equipe**, **#id\_stade**, **#id\_createur**, commentaire).

# 6 - Dictionnaire de données

| **Championnat** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id du championnat |
| intitule | intitulé du championnat |
| niveau | niveau du championnat (D1, D2 etc) |

| **Equipe** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id de l'équipe |
| nom | nom de l'équipe |
| ville | ville dans laquelle est établie l'équipe |
| id\_stade | id du stade de l'équipe |
| couleur | couleur du maillot principal de l'équipe |

| **participerChampionnat** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id\_equipe | id de l'équipe participant |
| id\_championnat | id du championnat auquel participe l'équipe |

| **Joueur** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| licence | numéro de la licence officiel de la FFF |
| nom | nom du joueur |
| prenom | prenom du joueur |
| id\_equipe | id de l'équipe du joueur |
| poste | poste principal dans lequel le joueur évolue |

| **Stade** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id du stade |
| nom | nom du stade |
| adresse | adresse à laquel se trouve le stade |
| code\_postal | code postal de la ville du stade |
| ville | ville du stade |
| type | type de terrain (herbes, stabilisé, synthétique) |

| **Match** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id du match |
| date | date du déroulement du match |
| heure\_debut | heure du début du match |
| heure\_fin | heure de fin du match |
| id\_equipe\_ext | id de l'équipe extérieur (visiteur) |
| id\_equipe\_dom | id de l'équipe domicile |
| id\_stade | id du stade où se déroule le match |
| commentaire | commentaire supplémentaire pour le match |

| **EvenementMatch** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id de l'événement |
| id\_match | id du match de l'évènement |
| licence\_joueur | licence du joueur concerné par l'évènement |
| type | type de l'événement (BUT, CARTON etc) |
| temps | le moment où l'événement s'est déclenché (en minute) |
| commentaire | Commentaire supplémentaire décrivant l'événement |
| licence\_passeur | Si il y'en a un la licence du passeur décisive |

| **EventCalendrier** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id de l'événement |
| intitulé | intitulé de l'évènement |
| heure\_deb | heure de début de l'événement |
| heure\_fin | heure de la fin de l'évènement |
| id\_equipe | id de l'équipe concernée |
| id\_stade | id du stade concerné |
| id\_createur | id du créateur de l'événement |
| commentaire | commentaire de l'évènement décrivant celui ci |

| **Inscrit** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id de l'utilisateur |
| licence | numéro de licence de l'utilisateur |
| mail | adresse mail de l'utilisateur |
| mdp | Mot de passe de l'utilisateur |

| **Message** | |
| --- | --- |
| **Attributs** | |
| **Nom** | **Description** |
| id | id du message |
| id\_auteur | id de l'utilisateur auteur du message |
| id\_destinataire | id de l'utilisateur qui reçoit le message |
| date | Date et heure de l'envoie du message |
| contenu | contenu du message |

# 7 - Manipulation de la Base de données

## 7.1 - Peuplement de la base de données

Afin de pouvoir utiliser l’application nous avions tout d’abord besoin d’avoir une base de données complète avec des données de joueurs, d’équipes, de stades, de championnats, de matchs. Malheureusement, après de nombreuses recherches, nous n’avons pas réussi à nous procurer une vraie base de données de la Fédération Française de Football. C’est pourquoi nous avons pris la décision de peupler notre propre base de données.

Afin d’avoir un jeu de données complet et efficace pour tester notre application, nous avons défini qu’une table avec minimum 3 championnats d’une dizaine d’équipe était nécessaire. Si on fait le calcule ça fait 30 équipes avec dedans une quinzaine de joueurs donc 30 \* 15 = 450, ça fait 450 joueurs à rentrer à la main dans la base de données, ça peut vite devenir incontrôlable et surtout très long.

Nous avons donc décidé de faire un script qui peuple automatiquement la base de données avec des données totalement fictives et aléatoires.

[Le programme de peuplement automatique](https://github.com/Arthur-Le-M/scriptBD_SportTrack)

Ce programme est écrit en Python et utilise plusieurs fichiers CSV libres d’utilisation afin de construire des données fictives mais cohérentes. Par exemple pour construire des données de joueurs aléatoires, nous avons besoin d’un nom et d’un prénom, pour cela nous avons trouvé sur le web un fichier CSV avec les nom de famille les plus utilisés en France et un autre fichier CSV avec les prénom les plus utilisés en France, dans chaque fichier il y a plusieurs milliers de données ce qui rends nos données finales uniques et cohérentes.

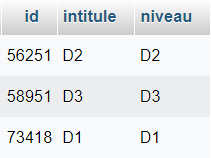
Le programme fonctionne de la façon suivante : le fichier “generateur.py” exploite les fichiers CSV afin d’en extraire et de construire des données aléatoire, le fichier “scriptTable.py” utilise les fonctions de generateur.py pour insérer des données dans la base de données :



# 

Le programme permet de remplir la base de données, avec des données qualitatives en quelques secondes, voici un échantillon d’une base de données peuplée grâce à ce programme :

**Championnat**



**Equipe**

**Joueur**



**Stade**



**participerChampionnat**



**Match**



**evenementMatch**



## 7.2 - Requêtes sur la base de données

Notre base de données est maintenant suffisamment peuplée pour que nous puissions utiliser notre application.

On va maintenant montrer des exemples de scénarii avec leurs requêtes associées :

**Inscription : 2 méthodes possibles**

1. Vérification si la licence existe :

SELECT licence FROM Joueur WHERE licence = licence;

Vérification si l’email et la licence sont utilisés :

SELECT \* FROM Inscrit WHERE licenceJoueur = licence OR mail = mail;

Insertion dans la BD :

INSERT INTO Inscrit (licence, mail, mdp)

VALUES (licence, mail, mdp);

2. Trigger pour vérification :

CREATE TRIGGER verif\_licence

BEFORE INSERT ON joueur

FOR EACH ROW

BEGIN

DECLARE nb\_licences INT;

SELECT COUNT(\*) INTO nb\_licences

FROM joueur

WHERE licence = licence;

IF nb\_licences > 0 THEN

INSERT INTO joueur (numero\_licence, nom, prenom,id\_equipe,poste)

VALUES (licence,nom,prenom,id\_equipe,poste);

ELSE

RAISEERROR('Le numéro de licence n’existe pas.');

END IF;

END;

**Envoyer un message**

Insertion du message dans la BD :

INSERT INTO Message (id\_auteur, id\_destinataire, date, contenue)

VALUES (id\_auteur, id\_destinataire, date, contenue);

**Supprimer un événement du calendrier**

Suppression de l'événement dans la BD :

DELETE FROM EvenementCalendrier WHERE id = id

**Consulter les message d’une conversation**

Sélection des messages dans l’ordre décroissant par rapport à la date :

SELECT contenu FROM Message

WHERE id\_auteur = id\_auteur AND id\_destinataire = id\_destinataire

ORDER BY date DESC;

**Afficher le nom de l’équipe à partir d’un joueur de celle ci**

SELECT Equipe.nom FROM Joueur

JOIN Equipe ON Joueur.id\_equipe = Equipe.id

WHERE Joueur.licence = licence

# 8 - Technologie utilisées

Pour utiliser la base de données, nous avons décidé d’utiliser l’application web PhpMyAdmin qui fonctionne avec le SGBD MySql puisque c’est un logiciel que nous avons utilisé en première année ainsi qu’en deuxième, donc nous le connaissons bien et nous le trouvons efficace pour ce qu’on a besoin de faire.

Enfin, pour pouvoir exploiter les données de la base sur notre futur site web **Sport Track**, nous utiliserons php puisque c’est un langage que nous avons appris lors du semestre 3. Nous avons donc acquis beaucoup de connaissances qui nous permettront de rendre notre site dynamique grâce à ce langage.

# 9. Consulter Scripts et base de données

URL de notre dossier Github : <https://github.com/Arthur-Le-M/scriptBD_SportTrack>

Vous retrouvez dans ce dossier, tous les scripts de peuplements de la base, les fichiers CSV contenant les données réelles et des algorithmes de génération de championnat.

Vous pouvez récupérer la base de données complète en suivant le guide inscrit sur le dossier Github.