Projet de mini-site sous Django : **classement de ligues de football**

I - Principe

Le site, permet de stocker et récupérer les informations suivantes :

- **Ligues** sportives (elles permettent le classement des **équipes ;** détail important : une équipe peut appartenir à plusieurs ligues et donc avoir plusieurs classement et scores**)**

  Une victoire donne 3 points, un match nul, 1, une défaite, 0. En cas d'égalité, selon une règle ancestrale, c'est l'ordre alphabétique inverse qui départage les équipes.

- **Equipes** ; elles appartiennent à plusieurs ligues et jouent des **Matchs**

- **...Matchs** Un match peut être passé (avec des scores) ou programmé dans le cadre d'une ligue (sans score). Dans ce dernier cas, un bouton permet de les générer automatiquement.

  Un match doit être rattaché à une ligue donnée (mais les équipes n'ont pas besoin d'appartenir à cette ligue)

II - Fonctionnement du site ("front")

1) Accueil

L'écran d'accueil */game/* affiche les ligues et les équipes relatives à ces ligues (en sachant qu'une équipe peut appartenir à plusieurs ligues)

A partir de l'écran d'accueil, on peut :

- Afficher le classement relatif à une ligue (lien hypertexte)
- Afficher le palmarès d'une équipe (toutes ligues confondues ; lien)
- Ajouter un Match (manuellement ; accessible via bouton)

2) Classement Ligue

Le classement par ligue se met à jour automatiquement lors du rafraichissement de la page (variable equipe.score2, relative à une ligue donnée, remplie par la méthode equipe.score())

Cette page permet également d'afficher les matchs déjà joué et de générer les matchs manquants (un match aller, un retour par couple d'équipe) via un bouton situé sous la liste des matchs

Après ajout d'un match à venir, le score peut être complété via un form invoqué en cliquant sur le score manquant, à condition que la date de match soit antérieure à maintenant. Une fois le score complété, il n'est plus possible de corriger sans console admin.

3) Palmarès équipe

Cet écran contient les matchs et les ligues auxquelles appartient l'équipe

4) Ajouter Match

Cet écran permet d'ajouter un match entièrement paramétré par l'utilisateur.

III - Caractéristiques techniques ("back")

1) Le modèle : 3 classes

- **Equipe** (nom, ligues (plusieurs possibles), score2, méthode "score", propriété "matches")
- **Ligue** (nom, propriété "equipes")
- **Match** (nom, date, locaux, visiteur, ligue, score_locaux, score_visiteurs)

2) Views : 3 vues, 3 fonctions

- **LigueView :** affiche la page d'acceuil ; utilise generic.ListView
- **EquipesView** : affiche le classement, utilise generic.ListView + surcharges de fonction, notamment pour le score2 qui est recalculé dans la base pour chaque affichage de ligue (une solution temporaire qui devra être améliorée)
- **EquipeView** : almarès d'une équipe précise ; utilise Generic.DetailView
- fonction **add_match :** permet d'ajouter un match via *add_match_form*

- fonction **create_matches :** génère les combinaisons de matchs aller et retour pour une ligue donnée
- fonction **modify_match :** permet de remplir le score d'un match planifié via **creat_matches**

En ce qui concerne la base de données, elle était au départ sur un serveur MySQL ; pour des raisons de livrable et de portabilité, je suis passé sur Sqlite3.

Pour la mise en forme des pages, j'ai utilisé les templates Django (dont extends pour la navbar) + bootstrap + quelques styles CSS.