Structure MVC du Projet

```
- main.py
|
- models/
|  |
|  - match.py
|  - player.py
|  - round.py
|  - tournament.py
|
- views/
|  |
|  - main_menu.py
|  - player.py
|  - report.py
|  - tournament.py
|  - view.py
|
- controller/
|  |
|  - database.py
|  - player.py
|  - tournament.py
|
- utils/
|  |
|  - check_date.py
|  - timestamp.py
|
- data/
|  |
|  - players.json
|  - tournaments.json
```

Le Dossier /models

```
Projet
|
|-- main.py
|
|-- models/
|   |
|   |-- match.py
|   |   |
|   |   |-- Class Match
|   |   |   |-- __init__(self, player_1, player_2, name)
|   |   |   |-- player_winner(self, winner)
|   |   |   |-- play_match(self)
|   |   |   |-- get_serialized_match(self)
|   |
|   |-- player.py
|   |   |
|   |   |-- Class Player
|   |   |   |-- __init__(self, name, firstname, birthday, gender, rating, to
|   |   |   |-- __str__(self)
|   |   |   |-- get_serialized_player(self)
|   |   |   |-- display_all_info(self)
|   |
|   |-- round.py
|   |   |
|   |   |-- Class Round
|   |   |   |-- __init__(self, name, players_pairs, load_match=False)
|   |   |   |-- __str__(self)
|   |   |   |-- create_matchs(self)
|   |   |   |-- mark_as_complete(self)
|   |   |   |-- get_serialized_round(self)
|   |
|   |-- tournament.py
|       |
|       |-- Class Tournament
|       |   |-- __init__(self, name, location, date, time_control, players,
|       |   |-- __str__(self)
|       |   |-- create_round(self, round_number)
|       |   |-- create_players_pairs(self, current_round)
|       |   |-- get_rankings(self, by_score=True)
|       |   |-- get_serialized_tournament(self, save_rounds=False)
|
```

Le Dossier /views

```
|-- views/
|   |
|   |-- main_menu.py
|   |   |
|   |   |-- Class MainMenu(View)
|   |   |   |-- display_main_menu(self)
|   |
|   |-- player.py
|   |   |
|   |   |-- Class CreatePlayer(View)
|   |   |   |-- display_menu(self)
|   |   |
|   |   |-- Class LoadPlayer(View)
|   |   |   |-- display_menu(self, nb_players_to_load)
|   |
|   |-- report.py
|   |   |
|   |   |-- Class Report(View)
|   |   |   |-- __init__(self)
|   |   |   |-- display_players_report(self, players=[])
|   |   |   |-- display_tournaments_reports(self)
|   |
|   |-- tournament.py
|   |   |
|   |   |-- Class CreateTournament(View)
|   |   |   |-- display_menu(self)
|   |   |
|   |   |-- Class LoadTournament(View)
|   |   |   |-- display_menu(self)
|   |
|   |-- view.py
|       |
|       |-- Class View
|       |   |-- @staticmethod
|       |   |   |-- get_user_entry(msg_display, msg_error, value_type, asser
|       |   |-- @staticmethod
|       |   |   |-- build_selection(iterable, display_msg, assertions)
```

Le Dossier /controller

```
|-- controller/
|   |
|   |-- database.py
|   |   |
|   |   |-- save_db(db_name, serialized_data)
|   |   |-- update_db(db_name, serialized_data)
|   |   |-- update_player_rank(db_name, serialized_data)
|   |   |-- load_db(db_name)
|   |   |-- load_player(serialized_player, load_tournament_score=False)
|   |   |-- load_tournament(serialized_tournament)
|   |   |-- load_rounds(serialized_tournament, tournament)
|   |   |-- load_match(serialized_match, tournament)
|   |
|   |-- player.py
|   |   |
|   |   |-- create_player()
|   |   |-- update_rankings(player, rank, score=True)
|   |   |-- display_players(sort_by="name", ascending=True)
|   |
|   |-- tournament.py
|       |
|       |-- create_tournament()
|       |-- play_tournament(tournament, new_tournament_loaded=False)
|
```

Les Dossiers /utils & /data

```
|-- utils/
|    |
|    |-- check_date.py
|    |    |
|    |    |-- check_date(date)
|    |
|    |-- timestamp.py
|         |
|         |-- get_timestamp()
|
|-- data/
|    |
|    |-- players.json
|    |-- tournaments.json
```

Division des 10 premières Étapes

Afin d'avoir une bonne approche pour la gestion de ce projet. J'ai divisé diviser les tâches sur **10 étapes :**

Conception du modèle de données : Cela comprendra la conception des classes et de leurs méthodes. Dans ce projet, cela concerne les classes de /models/ (match.py, player.py, round.py, tournament.py).

Création de la base de données et des opérations de la base de données : Cela concerne le dossier /controller/, spécifiquement database.py, où vous définirez comment sauvegarder, mettre à jour et charger les données dans votre base de données.

Gestion des joueurs : Développement de toutes les fonctions concernant la création et la gestion des joueurs. Cela comprendrait le travail sur player.py dans /models/, /views/ et /controller/.

Gestion des tournois : Développement de toutes les fonctionnalités liées à la création et à la gestion des tournois. Cela comprendrait le travail sur tournament.py dans /models/, /views/ et /controller/.

Développement des vues : Développement des différentes interfaces utilisateur dans /views/, telles que main_menu.py, report.py, et les autres fichiers .py dans le dossier.

Création des utilitaires : Cela comprendrait le développement des fichiers dans le dossier /utils/ comme check_date.py et timestamp.py.

Intégration et tests unitaires : Intégrer les différentes parties du projet ensemble et s'assurer que chaque unité fonctionne correctement.

Tests d'intégration : Tester le système dans son ensemble pour s'assurer que toutes les parties fonctionnent ensemble correctement.

Correction de bugs et optimisation : Après avoir effectué des tests, il y aura probablement des bugs à corriger et des parties du code à optimiser.

Documentation : Écrire la documentation du projet, y compris comment installer et utiliser le système, ainsi que des commentaires sur le code pour aider à sa maintenance.

Division et Regroupement de ces 10 Étapes en un Planning de 20 Taches :

Voici un plan de 20 étapes pour le développement du projet en respectant la cohérence et la logique de l'organisation du code.

1. **Définir les exigences du projet et la conception du système :** Compréhension des exigences du projet pour concevoir un système qui répond aux exigences. Cela comprend la planification des fonctionnalités du système, des classes et des méthodes que nous avons utilisés.

2. **Configuration de l'environnement de développement :** Configuration de l'environnement Python, choix d'un IDE, et configuration de tout autre outil nécessaire.

3. **Création de l'architecture de base du projet :** Créer les dossiers et les fichiers de base comme indiqué dans la structure du projet MVC.

4. **Développement du modèle Player :** Cela comprend le développement de la classe Player dans player.py sous le dossier /models/.

5. **Développement du modèle Match :** Cela comprend le développement de la classe Match dans match.py sous le dossier /models/.

6. **Développement du modèle Round :** Cela comprend le développement de la classe Round dans round.py sous le dossier /models/.

7. **Développement du modèle Tournament :** Cela comprend le développement de la classe Tournament dans tournament.py sous le dossier /models/.

8. **Création de la classe de base View :** Développer la classe de base View qui sera héritée par les autres classes de vue.

9. **Développement de la vue MainMenu :** Cela comprend le développement de la classe MainMenu dans main_menu.py sous le dossier /views/.

10. **Développement des vues Player :** Cela comprend le développement des classes CreatePlayer et LoadPlayer dans player.py sous le dossier /views/.

11. **Développement des vues Tournament :** Cela comprend le développement des classes CreateTournament et LoadTournament dans tournament.py sous le dossier /views/.

12. **Développement de la vue Report :** Cela comprend le développement de la classe Report dans report.py sous le dossier /views/.

13. **Développement des contrôleurs Player et Tournament :** Cela comprend le développement des fonctions dans player.py et tournament.py sous le dossier /controller/.

14. **Développement du contrôleur Database :** Cela comprend le développement des fonctions dans database.py sous le dossier /controller/.

15. **Développement des utilitaires :** Cela comprend le développement des fonctions dans check_date.py et timestamp.py sous le dossier /utils/.

16. **Intégration du code et développement du fichier main.py** : Cela comprend l'intégration de tous les fichiers de code et le développement du fichier main.py.

17. **Tests unitaires :** Tester chaque classe et fonction individuellement pour s'assurer qu'elles fonctionnent correctement.

18. **Tests d'intégration :** Tester le système dans son ensemble pour s'assurer que toutes les parties fonctionnent ensemble correctement.

19. **Correction de bugs et optimisation :** Après avoir effectué des tests, il y aura probablement des bugs à corriger et des parties du code à optimiser.

20. **Documentation et finalisation du projet :** Écrire la documentation du projet, y compris comment installer et utiliser le système, ainsi que des commentaires sur le code pour aider à sa maintenance. Finaliser le projet en le revoyant une dernière fois pour s'assurer que tout est en place.

Structure de mon projet :

- main.py
|
- models/
| |
| - match.py
| - player.py
| - round.py
| - tournament.py
|
- views/
| |
| - main_menu.py
| - player.py
| - report.py
| - tournament.py
| - view.py
|
- controller/
| |
| - database.py
| - player.py
| - tournament.py
|
- utils/
| |
| - check_date.py
| - timestamp.py
|
- data/
| |
| - players.json
| - tournaments.json

Toutes les classes et fonctions & méthodes de mon projet

 # Dossier : /models/

Fichier : /models/match.py

Class Match :

def __init__(self, player_1, player_2, name):

def player_winner(self, winner):

def play_match(self):

def get_serialized_match(self):

Fichier : /models/player.py

class Player:

def __init__(self, name, firstname, birthday, gender, rating, total_score):

def __str__(self):

def get_serialized_player(self):

def display_all_info(self):

Fichier : /models/round.py

class Round:
def __init__(self, name, players_pairs, load_match=False):

def __str__(self):

def create_matchs(self):

def mark_as_complete(self):

def get_serialized_round(self):

Fichier : /models/tournament.py

class Tournament:
def __init__(self, name, location, date, time_control, players,
rounds_number=ROUNDS_NUMBER, description=""):

def __str__(self):

def create_round(self, round_number):
def create_players_pairs(self, current_round):

def get_rankings(self, by_score=True):

def get_serialized_tournament(self, save_rounds=False):

 # Dossier : /views/

Fichier : /views/main_menu.py

class MainMenu(View):
def display_main_menu(self):

Fichier : /views/player.py

class CreatePlayer(View):
def display_menu(self):

class LoadPlayer(View):
def display_menu(self, nb_players_to_load):

Fichier : /views/report.py

class Report(View):
    def __init__(self):

    def display_players_report(self, players=[]):

     def display_tournaments_reports(self):

Fichier : /views/tournament.py

class CreateTournament(View):
    def display_menu(self):

class LoadTournament(View):
    def display_menu(self):

Fichier : /views/view.py

class View:
    @staticmethod
    def get_user_entry(msg_display, msg_error, value_type, assertions=None, default_value=None):

    @staticmethod
    def build_selection(iterable, display_msg, assertions):

 # Dossier : /controller/

Fichier : /controller/database.py
    def save_db(db_name, serialized_data):

    def update_db(db_name, serialized_data):

    def update_player_rank(db_name, serialized_data):

    def load_db(db_name):

    def load_player(serialized_player, load_tournament_score=False):

    def load_tournament(serialized_tournament):

    def load_rounds(serialized_tournament, tournament):

    def load_match(serialized_match, tournament):

Fichier : /controller/player.py
    def create_player():

    def update_rankings(player, rank, score=True):

    def display_players(sort_by="name", ascending=True):

Fichier : /controller/tournament.py
    def create_tournament():

def play_tournament(tournament, new_tournament_loaded=False):

 # Dossier : /utils/

Fichier : /utils/chek_date.py

def check_date(date):

Fichier : /utils/timestamp.py

def get_timestamp():

 # Dossier : /data/

Fichier : /data/players.json

Fichier : /data/tournaments.json

A La Racine du Projet :

Fichier : /main.py