Certamente! Prima di procedere con il TeamTieringService, facciamo un riepilogo sintetico delle modifiche e poi prepariamo la documentazione per i comandi Artisan che abbiamo creato e modificato.

**Fix Sintetico (Commit Message Suggerito):**

feat(data-import): Enhance team data import and historical standings management  
  
- Implement `TeamsImportStandingsFile` Artisan command to import historical league standings from CSV files.  
 - Add `--create-missing-teams` option to automatically create team records if not found in DB during CSV import.  
 - Add options `--default-tier-for-new` and `--is-serie-a-league` for newly created teams.  
 - Improve CSV parsing by specifying delimiter and enhancing error logging.  
- Enhance `MapTeamApiIdsCommand` Artisan command:  
 - Add `--competition` option to map API IDs for different leagues (e.g., Serie A, Serie B).  
 - Improve name normalization and matching logic for team records.  
 - Add detailed logging for matching process.  
- Enhance `TeamDataService`:  
 - Modify `fetchAndStoreSeasonStandings` to accept a `competitionId` parameter.  
 - Improve team matching logic by prioritizing API ID, then normalized names/shortnames/TLAs.  
 - Implement auto-update of `api\_football\_data\_team\_id` in local `teams` table if a match is found by name and ID API is missing/different.  
 - Refine cache duration for API responses (shorter for recent, longer for historical).  
 - Improve error handling for API responses, including 403 errors.  
- Database: `team\_historical\_standings` table schema finalized.  
- Config: Add `team\_tiering\_settings.api\_football\_data.serie\_a\_competition\_id` and `standings\_endpoint`. Add cache TTLs to `cache.php`. Add `api\_delay\_seconds` to `services.php`.  
  
This commit provides robust data ingestion capabilities for historical team performance,  
setting the stage for dynamic team tiering.

## **Documentazione Comandi Artisan Personalizzati per FantaProject**

Questo documento descrive i comandi Artisan personalizzati creati per l'applicazione FantaProject, con spiegazioni sul loro scopo, utilizzo, opzioni disponibili, esempi e l'ordine di esecuzione consigliato per la gestione dei dati.

### **1. test:projection {playerId}**

* **Spiegazione:** Testa il ProjectionEngineService per un singolo giocatore specifico, generando e visualizzando le sue proiezioni statistiche e di FantaMedia per la stagione. Utilizza il primo UserLeagueProfile trovato nel database o ne crea uno di default se non esiste.
* **Utilizzo:** Utile per il debug della logica di proiezione, per verificare l'impatto di modifiche ai parametri di configurazione (config/projection\_settings.php, config/player\_age\_curves.php) su un giocatore campione, o per analizzare rapidamente le aspettative per un singolo calciatore.
* **Argomenti:**
  + playerId: (Obbligatorio) Il fanta\_platform\_id del giocatore da testare.
* **Esempio:**  
  Bash  
  php artisan test:projection 2170  
  (Dove 2170 è l'ID di V. Milinkovic-Savic, come da esempi precedenti)
* **File Coinvolti:** app/Console/Commands/TestPlayerProjectionCommand.php, app/Services/ProjectionEngineService.php, app/Services/FantasyPointCalculatorService.php, modelli Player, UserLeagueProfile, Team, HistoricalPlayerStat.

### **2. players:enrich-data**

* **Spiegazione:** Arricchisce i dati dei giocatori presenti nella tabella players interrogando l'API esterna Football-Data.org. Recupera informazioni come la data di nascita (date\_of\_birth), la posizione dettagliata (detailed\_position) e l'ID univoco del giocatore sull'API (api\_football\_data\_id).
* **Utilizzo:** Fondamentale per ottenere dati anagrafici accurati (specialmente l'età, usata dal ProjectionEngineService). Da eseguire dopo l'importazione iniziale del roster e periodicamente se si aggiungono nuovi giocatori.
* **Opzioni:**
  + --player\_id=all|ID: (Default: all) Specifica se arricchire tutti i giocatori che necessitano di dati o un giocatore specifico tramite il suo ID del database locale.
  + --player\_name=NOME: Arricchisce i giocatori il cui nome contiene la stringa specificata.
  + --delay=SECONDI: (Default: 6) Numero di secondi di attesa tra le chiamate API per giocatore, per rispettare i rate limit.
* **Esempi:**  
  Bash  
  php artisan players:enrich-data   
  php artisan players:enrich-data --player\_name="osimhen"  
  php artisan players:enrich-data --player\_id=123 --delay=10
* **File Coinvolti:** app/Console/Commands/EnrichPlayerDataCommand.php, app/Services/DataEnrichmentService.php, modello Player, config/services.php (per API key).

### **3. teams:map-api-ids**

* **Spiegazione:** Associa le squadre presenti nel database locale (tabella teams) con i loro ID corrispondenti dall'API Football-Data.org. Popola o aggiorna il campo api\_football\_data\_team\_id nella tabella teams.
* **Utilizzo:** Cruciale per permettere al TeamDataService di identificare correttamente le squadre quando scarica le classifiche storiche via API. Da eseguire dopo aver popolato la tabella teams (es. con TeamSeeder.php) e ogni volta che si aggiungono nuove squadre al sistema che potrebbero necessitare di un mapping API.
* **Opzioni:**
  + --season=YYYY: (Opzionale) Anno di inizio stagione. L'endpoint API /teams di una competizione potrebbe restituire le squadre di una stagione specifica o quelle della stagione corrente definita dall'API.
  + --competition=CODICE\_LEGA: (Default: SA) Il codice della competizione per cui mappare le squadre (es. SA per Serie A, SB per Serie B, se supportato dall'API e dal tuo piano).
* **Esempi:**  
  Bash  
  php artisan teams:map-api-ids --competition=SA  
  php artisan teams:map-api-ids --competition=SB --season=2023
* **File Coinvolti:** app/Console/Commands/MapTeamApiIdsCommand.php (come da codice fornito), modello Team, config/services.php, config/team\_tiering\_settings.php (per serie\_a\_competition\_id).

### **4. teams:fetch-historical-standings**

* **Spiegazione:** Recupera i dati storici delle classifiche per una specifica competizione e stagione (o più stagioni recenti) dall'API Football-Data.org e li salva nella tabella team\_historical\_standings. Tenta di mappare le squadre API ai team locali usando api\_football\_data\_team\_id o, in fallback, il nome normalizzato. Se trova una corrispondenza per nome e l'api\_football\_data\_team\_id locale è mancante/diverso, lo aggiorna.
* **Utilizzo:** Per popolare automaticamente lo storico delle classifiche, necessario per il TeamTieringService. Da eseguire dopo teams:map-api-ids per massimizzare le corrispondenze.
* **Opzioni:**
  + --season=YYYY: (Opzionale) Anno di inizio stagione specifico da scaricare (es. 2023 per la stagione 2023-24).
  + --all-recent=N: (Opzionale) Scarica le classifiche per le ultime N stagioni recenti.
  + --competition=CODICE\_LEGA: (Default: SA) Il codice della competizione (es. SA, SB).
* **Esempi:**  
  Bash  
  php artisan teams:fetch-historical-standings --season=2023 --competition=SA  
  php artisan teams:fetch-historical-standings --all-recent=3 --competition=SA  
  php artisan teams:fetch-historical-standings --season=2023 --competition=SB
* **File Coinvolti:** app/Console/Commands/TeamsFetchHistoricalStandings.php (come da codice fornito), app/Services/TeamDataService.php (come da codice fornito), modelli Team, TeamHistoricalStanding, config/services.php, config/team\_tiering\_settings.php, config/cache.php.

### **5. teams:import-standings-file**

* **Spiegazione:** Importa i dati storici delle classifiche da un file CSV locale nella tabella team\_historical\_standings. Permette di creare automaticamente nella tabella teams le squadre presenti nel CSV ma non nel database.
* **Utilizzo:** Fondamentale per popolare lo storico delle classifiche per stagioni/leghe non accessibili tramite API (a causa di restrizioni del piano o indisponibilità dei dati) o per un setup iniziale massivo.
* **Argomenti:**
  + filepath: (Obbligatorio) Percorso al file CSV da importare.
* **Opzioni:**
  + --season-start-year=YYYY: (Obbligatorio) Anno di inizio della stagione a cui si riferiscono i dati nel CSV (es. 2021 per la stagione 2021-22).
  + --league-name="Nome Lega": (Default: Serie A) Nome della lega per i dati importati (es. "Serie A", "Serie B").
  + --create-missing-teams=true|false: (Default: false) Se true, crea un record nella tabella teams se una squadra nel CSV non viene trovata.
  + --default-tier-for-new=TIER: (Default: 4) Tier da assegnare alle squadre create con --create-missing-teams=true.
  + --is-serie-a-league=true|false: (Default: true) Imposta il flag serie\_a\_team per le squadre create. Usare false per leghe come la Serie B.
* **Esempi:**  
  Bash  
  php artisan teams:import-standings-file storage/app/import/classifica\_serie\_a\_2021-22.csv --season-start-year=2021  
  php artisan teams:import-standings-file storage/app/import/classifica\_serie\_b\_2022-23.csv --season-start-year=2022 --league-name="Serie B" --create-missing-teams=true --default-tier-for-new=5 --is-serie-a-league=false
* **File Coinvolti:** app/Console/Commands/TeamsImportStandingsFile.php (come da codice fornito), modelli Team, TeamHistoricalStanding. Richiede league/csv (o alternativa per parsing CSV/XLSX).

**Ordine di Esecuzione Consigliato (Workflow Tipico per Setup e Manutenzione):**

1. **Setup Iniziale dell'Applicazione:**
   * php artisan migrate (per creare le tabelle).
   * php artisan db:seed (per eseguire il TeamSeeder.php e altri seeder).
   * php artisan teams:map-api-ids --competition=SA (per mappare gli ID API delle squadre di Serie A già presenti).
   * php artisan teams:map-api-ids --competition=SB (se vuoi mappare squadre di Serie B già presenti e l'API lo permette).
   * php artisan players:enrich-data (per popolare date di nascita, etc., dei giocatori importati con il roster).
2. **Popolamento Storico Classifiche (una tantum o per stagioni non API):**
   * Per ogni stagione/lega per cui hai un file CSV: php artisan teams:import-standings-file path/al/tuo.csv --season-start-year=YYYY --league-name="Nome Lega" --create-missing-teams=true ...
   * Dopo aver importato CSV che potrebbero aver creato nuovi team che erano in Serie A (o lo saranno), riesegui: php artisan teams:map-api-ids --competition=SA (per tentare di mappare gli ID API ai nuovi team).
3. **Download Storico Classifiche via API (per stagioni accessibili):**
   * php artisan teams:fetch-historical-standings --all-recent=3 --competition=SA (per le ultime 3 stagioni di Serie A).
   * php artisan teams:fetch-historical-standings --season=YYYY --competition=SA (per stagioni specifiche).
   * Eventualmente per la Serie B se l'accesso API è disponibile: php artisan teams:fetch-historical-standings --all-recent=2 --competition=SB
4. **Aggiornamento Tier Squadre (inizio nuova stagione di proiezione):**
   * (Fase Precedente) Assicurati che il flag serie\_a\_team nella tabella teams sia corretto per le squadre che parteciperanno alla Serie A della stagione per cui vuoi calcolare i tier (vedi discussione su teams:set-active-league).
   * php artisan teams:update-tiers YYYY-YY (es. 2024-25) -> *Questo comando lo svilupperemo come prossimo passo*.
5. **Generazione Proiezioni Giocatori:**
   * Dopo che i tier sono aggiornati, le proiezioni dei giocatori (es. tramite test:projection o future interfacce web) useranno i tier più recenti.

Questo flusso dovrebbe garantire che i tuoi dati siano il più possibile completi e aggiornati prima di calcolare i tier delle squadre e, successivamente, le proiezioni dei giocatori.