**8.5. Comandi Artisan Personalizzati (Flusso Operativo e Dettagli)**

Questa sezione descrive i comandi Artisan personalizzati sviluppati per l'applicazione FantaProject. Questi comandi sono essenziali per l'importazione dei dati, l'arricchimento, il calcolo dei tier delle squadre e il testing delle proiezioni.

**Flusso Operativo Consigliato per Inizio Nuova Stagione (es. per proiezioni stagione 2025-26):**

1. **Aggiornamento Dati Storici Stagione Appena Conclusa (2024-25):**
   * Se disponibile via API: php artisan teams:fetch-historical-standings --season=2024 --competition=SA
   * Per leghe/stagioni non API (es. Serie B 24-25): php artisan teams:import-standings-file storage/app/import/classifica\_serie\_b\_2024-25.csv --season-start-year=2024 --league-name="Serie B" --create-missing-teams=true --is-serie-a-league=false
2. **Definizione Squadre Attive per la Nuova Stagione (2025-26):**
   * php artisan teams:set-active-league --target-season-start-year=2025 --league-code=SA --set-inactive-first=true
3. **Mappatura ID API Squadre (Opzionale, se non già coperto dai passi precedenti):**
   * php artisan teams:map-api-ids --competition=SA (per assicurarsi che tutte le squadre attive abbiano l'ID API)
4. **Calcolo Tier Squadre per la Nuova Stagione (2025-26):**
   * php artisan teams:update-tiers 2025-26
   * *(Iterare questo passo con modifiche a config/team\_tiering\_settings.php se necessario per calibrare)*
5. **Importazione Nuovo Roster Giocatori (2025-26):**
   * Tramite interfaccia web (/uploads/roster).
6. **Importazione Nuove Statistiche Storiche Giocatori (se disponibili per stagione 24-25):**
   * Tramite interfaccia web (/uploads/historical-stats).
7. **Arricchimento Dati Giocatori (specialmente per nuovi arrivati):**
   * php artisan players:enrich-data
8. **Testing Proiezioni:**
   * php artisan test:projection {playerId} (per giocatori specifici)

**Dettaglio Comandi Artisan:**

**1. players:enrich-data**

* **Scopo:** Arricchisce i dati dei giocatori presenti nella tabella players interrogando l'API esterna Football-Data.org. Recupera informazioni cruciali 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). Questi dati sono fondamentali per il calcolo dell'età (usato nelle curve di crescita/declino) e per un corretto matching con altre fonti dati.
* **File Sorgente:** app/Console/Commands/EnrichPlayerDataCommand.php
* **Servizio Utilizzato:** App\Services\DataEnrichmentService
* **Utilizzo Consigliato:** Da eseguire dopo l'importazione iniziale del roster e ogni volta che vengono aggiunti nuovi giocatori al database (es. tramite import CSV di classifiche o roster aggiornati) per cui mancano i dati anagrafici.
* **Opzioni:**
  + --player\_id=<ID\_DB>: Arricchisce un giocatore specifico tramite il suo ID del database locale (players.id). Se omesso o all, processa tutti i giocatori che necessitano di arricchimento (es. date\_of\_birth nullo).
  + --player\_name="<NOME\_GIOCATORE>": Cerca e arricchisce i giocatori il cui nome (players.name) contiene la stringa specificata. Utile per targettizzare un giocatore specifico o un piccolo gruppo.
  + --delay=<SECONDI>: (Default: 6, configurabile in config/services.php sotto football\_data.api\_delay\_seconds) Numero di secondi di attesa tra le chiamate API per giocatore, per rispettare i rate limit dell'API.
  + --force-update=true|false: (Default: false) Se true, tenta di aggiornare i dati anche per i giocatori che li hanno già.
* **Esempi:**
  + Arricchire tutti i giocatori che ne necessitano:  
    Bash  
    php artisan players:enrich-data
  + Arricchire un giocatore specifico per nome:  
    Bash  
    php artisan players:enrich-data --player\_name="Osimhen"
  + Arricchire un giocatore specifico per ID DB, forzando l'aggiornamento e con un delay maggiore:  
    Bash  
    php artisan players:enrich-data --player\_id=123 --force-update=true --delay=10

**2. teams:set-active-league**

* **Scopo:** Definisce quali squadre sono considerate partecipanti attive a una specifica lega (primariamente la Serie A) per una stagione target. Aggiorna il flag booleano serie\_a\_team nella tabella teams. Recupera la lista dei team partecipanti direttamente dall'API Football-Data.org per la stagione e lega specificate.
* **File Sorgente:** app/Console/Commands/TeamsSetActiveLeague.php
* **Utilizzo Consigliato:** Eseguire all'inizio della preparazione per una nuova stagione di proiezione, prima di calcolare i tier delle squadre (teams:update-tiers). Assicura che la normalizzazione dei punteggi e l'assegnazione dei tier avvengano solo sul corretto pool di squadre.
* **Opzioni:**
  + --target-season-start-year=<ANNO>: (Obbligatorio) Anno di inizio della stagione per cui definire le squadre attive (es. 2024 per la stagione 2024-25).
  + --league-code=<CODICE\_LEGA>: (Default: SA) Il codice della competizione (es. SA per Serie A, SB per Serie B, come definito dall'API e in config/team\_tiering\_settings.php). Attualmente, il flag DB aggiornato è primariamente serie\_a\_team.
  + --set-inactive-first=true|false: (Default: true) Se true, imposta serie\_a\_team = false (o il flag corrispondente alla lega) per tutte le squadre prima di attivare quelle ricevute dall'API. È il metodo consigliato per una pulizia corretta.
* **Esempio:**
  + Impostare le squadre attive per la Serie A per la stagione che inizia nel 2024 (stagione 2024-25):  
    Bash  
    php artisan teams:set-active-league --target-season-start-year=2024 --league-code=SA
* **Note:** Il comando tenta di fare un match delle squadre API prima per api\_football\_data\_team\_id e poi per nome normalizzato. Se un team API non viene trovato nel DB locale, viene loggato un warning.

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

* **Scopo:** 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. Questo ID è cruciale per un matching affidabile con i dati provenienti dall'API (es. classifiche).
* **File Sorgente:** app/Console/Commands/MapTeamApiIdsCommand.php
* **Utilizzo Consigliato:** Eseguire dopo aver popolato inizialmente la tabella teams (es. con TeamSeeder.php) e ogni volta che si aggiungono nuove squadre al sistema (es. tramite import CSV) che potrebbero necessitare di un mapping API. Può essere utile eseguirlo dopo teams:set-active-league per assicurarsi che tutti i team attivi abbiano l'ID.
* **Opzioni:**
  + --competition=<CODICE\_LEGA>: (Default: SA, preso da config/team\_tiering\_settings.php) Il codice della competizione API per cui mappare le squadre (es. SA per Serie A, SB per Serie B, se supportato dall'API e dal tuo piano).
  + --season=<ANNO>: (Opzionale) Anno di inizio stagione. L'endpoint API (competitions/{id}/teams) potrebbe restituire le squadre di una stagione specifica o quelle della stagione corrente definita dall'API. Se omesso, usa l'endpoint senza filtro stagione.
* **Esempi:**
  + Mappare ID API per le squadre della Serie A (stagione corrente API):  
    Bash  
    php artisan teams:map-api-ids --competition=SA
  + Mappare ID API per le squadre della Serie B per la stagione 2023-24 (se l'endpoint lo supporta):  
    Bash  
    php artisan teams:map-api-ids --competition=SB --season=2023

**4. teams:import-standings-file**

* **Scopo:** 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 ancora presenti nel database.
* **File Sorgente:** app/Console/Commands/TeamsImportStandingsFile.php
* **Utilizzo Consigliato:** Fondamentale per popolare lo storico delle classifiche per stagioni o leghe non accessibili tramite API (a causa di restrizioni del piano, indisponibilità dei dati API per stagioni remote) o per un bootstrap iniziale massivo di dati storici.
* **Argomenti:**
  + filepath: (Obbligatorio) Percorso relativo alla root del progetto del file CSV da importare (es. storage/app/import/classifica\_serie\_a\_2020-21.csv).
* **Opzioni:**
  + --season-start-year=<ANNO>: (Obbligatorio) Anno di inizio della stagione a cui si riferiscono i dati nel CSV (es. 2020 per la stagione 2020-21).
  + --league-name="<NOME\_LEGA>": (Default: Serie A) Nome della lega per i dati importati (es. "Serie A", "Serie B"). Questo viene salvato in team\_historical\_standings.league\_name.
  + --create-missing-teams=true|false: (Default: false) Se true, crea un record nella tabella teams se una squadra nominata nel CSV non viene trovata per nome/short\_name.
  + --default-tier-for-new=<TIER>: (Default: 4) Tier da assegnare alle squadre create "al volo" se --create-missing-teams=true.
  + --is-serie-a-league=true|false: (Default: true) Quando si creano nuovi team (--create-missing-teams=true), imposta il flag serie\_a\_team a questo valore. Usare false per leghe come la Serie B.
* **Formato CSV Atteso:** Colonne come Posizione, Squadra, Punti, G, V, N, P, GF, GS, DR. Il comando ha una mappatura interna per questi nomi di colonna comuni.
* **Esempi:**
  + Importare la Serie A 2020-21, creando team mancanti:  
    Bash  
    php artisan teams:import-standings-file storage/app/import/serie\_a\_20-21.csv --season-start-year=2020 --create-missing-teams=true
  + Importare la Serie B 2022-23, creando team mancanti e specificando che non sono di Serie A:  
    Bash  
    php artisan teams:import-standings-file storage/app/import/serie\_b\_22-23.csv --season-start-year=2022 --league-name="Serie B" --create-missing-teams=true --default-tier-for-new=5 --is-serie-a-league=false

**5. teams:fetch-historical-standings**

* **Scopo:** 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 o diverso, lo aggiorna.
* **File Sorgente:** app/Console/Commands/TeamsFetchHistoricalStandings.php
* **Servizio Utilizzato:** App\Services\TeamDataService
* **Utilizzo Consigliato:** Per popolare/aggiornare automaticamente lo storico delle classifiche per le stagioni e leghe accessibili tramite API. Eseguire dopo teams:map-api-ids per massimizzare le corrispondenze via ID API.
* **Opzioni:**
  + --season=<ANNO>: (Opzionale) Anno di inizio stagione specifico da scaricare (es. 2023 per la stagione 2023-24).
  + --all-recent=<NUMERO>: (Opzionale) Scarica le classifiche per le ultime <NUMERO> stagioni recenti, partendo dalla stagione precedente a quella corrente.
  + --competition=<CODICE\_LEGA>: (Default: SA, preso da config/team\_tiering\_settings.php) Il codice della competizione API (es. SA per Serie A, SB per Serie B).
* **Esempi:**
  + Scaricare la classifica della Serie A per la stagione 2023-24:  
    Bash  
    php artisan teams:fetch-historical-standings --season=2023 --competition=SA
  + Scaricare le classifiche delle ultime 3 stagioni di Serie A:  
    Bash  
    php artisan teams:fetch-historical-standings --all-recent=3 --competition=SA
* **Note:** Utilizza la cache per le risposte API, con TTL configurabili in config/cache.php (ttl\_api\_standings\_recent\_hours, ttl\_api\_standings\_historical\_days).

**6. teams:update-tiers**

* **Scopo:** Esegue il TeamTieringService per ricalcolare e aggiornare il campo tier nella tabella teams per tutte le squadre marcate come attive (es. serie\_a\_team = true) per la stagione specificata.
* **File Sorgente:** app/Console/Commands/TeamsUpdateTiers.php
* **Servizio Utilizzato:** App\Services\TeamTieringService
* **Utilizzo Consigliato:** Eseguire dopo aver aggiornato i dati storici delle classifiche (team\_historical\_standings) e dopo aver definito le squadre attive per la stagione target (tramite teams:set-active-league). Questo passo è cruciale per preparare i tier corretti che verranno usati dal ProjectionEngineService.
* **Argomenti:**
  + targetSeasonYear: (Obbligatorio) La stagione **PER CUI** calcolare i tier, nel formato "YYYY-YY" (es. "2024-25"). Il servizio guarderà alle stagioni *precedenti* a questa per i suoi calcoli.
* **Esempio:**
  + Calcolare e aggiornare i tier per la stagione 2024-25:  
    Bash  
    php artisan teams:update-tiers 2024-25
* **Output:** Mostra il numero di squadre aggiornate, la nuova distribuzione dei tier e una tabella con i punteggi forza grezzi e normalizzati calcolati per le squadre processate.
* **Configurazione:** Il comportamento del servizio è pesantemente influenzato da config/team\_tiering\_settings.php.

**7. test:projection**

* **Scopo:** 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. Fa uso dei tier squadra più recenti disponibili.
* **File Sorgente:** app/Console/Commands/TestPlayerProjectionCommand.php
* **Utilizzo Consigliato:** 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, config/team\_tiering\_settings.php) su un giocatore campione, o per analizzare rapidamente le aspettative per un singolo calciatore dopo aver aggiornato i tier.
* **Argomenti:**
  + playerId: (Obbligatorio) Il fanta\_platform\_id del giocatore da testare (come presente nella tabella players).
* **Esempio:**
  + Testare le proiezioni per il giocatore con fanta\_platform\_id = 2170:  
    Bash  
    php artisan test:projection 2170