Comprendo perfettamente! Vuoi un documento "Fanta-Project Tecnico v1.3" completo, in formato Markdown come i precedenti, che integri tutte le nuove funzionalità relative al tiering dinamico, ai nuovi comandi Artisan e ai servizi associati.

Mi baserò sulla struttura del documento "Fanta-asta v1.2.1.docx" e integrerò tutte le novità che abbiamo discusso e implementato.

Ecco la bozza completa. Ho cercato di essere il più dettagliato possibile, specialmente nelle nuove sezioni e nella descrizione dei comandi Artisan.

**Documento Tecnico: Applicazione Laravel per Proiezioni e Strategia Asta Fantacalcio**

Versione: 1.3

Data: 30 maggio 2025

**Storico Revisioni:**

* Versione 1.0: Stesura iniziale.
* Versione 1.1: Aggiornamenti a Database, Servizi e Modulo Proiezioni.
* Versione 1.2: Integrazione API esterna (Football-Data.org), arricchimento dati giocatore (età, posizione dettagliata), implementazione aggiustamento per età nelle proiezioni, comando Artisan per arricchimento dati.
* **Versione 1.3 (Attuale):** Implementazione importazione dati storici classifiche squadre (API e CSV), sviluppo TeamDataService, implementazione sistema di Tiering Dinamico Squadre (TeamTieringService, config/team\_tiering\_settings.php), nuovi comandi Artisan per gestione dati squadre e tier, affinamento ProjectionEngineService per contributo clean sheet e uso tier dinamici, comando per gestione squadre attive per lega.

**Indice:**

1. Obiettivo del Progetto
2. Architettura di Sistema e Tecnologie
3. Gestione dei Dati 3.1. Fonte Dati Primaria (Input Utente) 3.2. Fonti Dati per Arricchimento (API Esterne) 3.3. Database Applicativo
4. Modulo di Tiering Dinamico delle Squadre 4.1. Scopo e Obiettivi del Tiering Dinamico 4.2. Fonte Dati per il Tiering 4.3. Logica di Calcolo del Punteggio Forza e Assegnazione Tier 4.4. Configurazione (config/team\_tiering\_settings.php) 4.5. Gestione Squadre Attive per Lega
5. Modulo di Proiezione Performance Calciatori 5.1. Dati di Input per le Proiezioni 5.2. Logica di Calcolo delle Proiezioni (Stato Attuale e Sviluppi) 5.3. Output delle Proiezioni
6. Modulo di Valutazione e Identificazione Talenti (Futuro)
7. Modulo Strategia d'Asta (Futuro)
8. Struttura Applicativa Laravel (Alto Livello) 8.1. Modelli Principali (Eloquent) 8.2. Servizi Chiave 8.3. Controller e Viste Principali 8.4. Processi in Background (Jobs) 8.5. Comandi Artisan Personalizzati (Flusso Operativo e Dettagli)
9. Considerazioni Aggiuntive e Sviluppi Futuri

1. Obiettivo del Progetto

L'obiettivo primario è sviluppare un'applicazione web basata su Laravel che assista l'utente nella preparazione e nella conduzione dell'asta del Fantacalcio (Serie A). L'applicazione fornirà proiezioni personalizzate sulle performance dei calciatori, identificherà giocatori sottovalutati (in futuro) e aiuterà a definire una strategia d'asta ottimale (in futuro), tenendo conto delle regole specifiche della lega dell'utente, della forza dinamicamente calcolata delle squadre e di dinamiche di mercato complesse.

**2. Architettura di Sistema e Tecnologie**

* **Piattaforma:** Applicazione Web
* **Framework Backend:** Laravel (PHP) - Versione corrente nel progetto
* **Database:** Database relazionale (configurato in config/database.php, es. MySQL)
* **Frontend:** Blade templates (files in resources/views/). JavaScript con resources/js/app.js e bootstrap.js (come da webpack.mix.js).
* **Librerie Chiave Utilizzate:**
  + Maatwebsite/Laravel-Excel (per importazione/esportazione XLSX).
  + League/Csv (per importazione CSV).
  + Laravel HTTP Client (basato su Guzzle) per chiamate API esterne.
  + Carbon (per manipolazione date/età).
* **Ambiente di Sviluppo Locale:** Laragon (come indicato dai path nei log).

**3. Gestione dei Dati**

**3.1. Fonte Dati Primaria (Input Utente)**

* **File XLSX Roster Ufficiale:** Caricamento tramite interfaccia web (gestito da RosterImportController). Contenuto: Lista calciatori, ruoli, quotazioni iniziali, ID piattaforma Fantacalcio.
* **File XLSX Statistiche Storiche Giocatori:** Caricamento tramite interfaccia web (gestito da HistoricalStatsImportController). Contenuto: Statistiche individuali per giocatore e stagione.
* **File CSV Classifiche Storiche Squadre (Opzionale/Fallback):** Importazione tramite comando Artisan (teams:import-standings-file) per stagioni/leghe non coperte dall'API. Contenuto: Posizione, punti, GF, GS, etc. per squadra e stagione.

**3.2. Fonti Dati per Arricchimento (API Esterne)**

* **API Utilizzata:** Football-Data.org (v4).
* **Configurazione API:** Chiave e URI base in .env e config/services.php.
* **Dati Giocatore Recuperati (tramite DataEnrichmentService):** Data di nascita, ID API giocatore, posizione dettagliata.
* **Dati Squadre Recuperati (tramite TeamDataService):**
  + Liste squadre per competizione e stagione (usate da teams:map-api-ids e teams:set-active-league).
  + Classifiche storiche per competizione e stagione (usate da teams:fetch-historical-standings per popolare team\_historical\_standings).
* **Dati Potenziali Futuri dall'API:** Nazionalità giocatori, statistiche avanzate (xG, xA).

**3.3. Database Applicativo**

* **players (App\Models\Player):** Anagrafica base, dati Fantacalcio, campi arricchiti da API (api\_football\_data\_id, date\_of\_birth, detailed\_position). Include team\_id (FK a teams).
* **teams (App\Models\Team):** name, short\_name, api\_football\_data\_team\_id (per mapping API), serie\_a\_team (boolean, per indicare partecipazione alla Serie A nella stagione target, gestito da teams:set-active-league), tier (integer, calcolato dinamicamente da TeamTieringService).
* **historical\_player\_stats (App\Models\HistoricalPlayerStat):** Statistiche individuali dei giocatori per stagione.
* **user\_league\_profiles (App\Models\UserLeagueProfile):** Configurazioni lega utente (budget, rose, regole di punteggio).
* **import\_logs (App\Models\ImportLog):** Tracciamento importazioni file.
* **team\_historical\_standings (App\Models\TeamHistoricalStanding):** Memorizza dati storici delle classifiche delle squadre (posizione, punti, gol, etc.) per stagione e lega. Popolata tramite API (TeamDataService / teams:fetch-historical-standings) o import CSV (teams:import-standings-file).
* **(Futuri)** player\_tactical\_notes, auction\_plans, etc.

**4. Modulo di Tiering Dinamico delle Squadre**

4.1. Scopo e Obiettivi del Tiering Dinamico

Lo scopo è superare un tiering statico delle squadre, fornendo una valutazione della forza di una squadra (tier) che sia:

* **Basata su Dati:** Calcolata analizzando le performance storiche.
* **Adattiva:** Riflette l'andamento recente e la forza relativa delle squadre.
* **Configurabile:** Permette di pesare diversamente le stagioni storiche e le metriche di performance.
* **Modulabile:** Considera la differenza di competitività tra diverse leghe (es. Serie A vs Serie B).

Questo tier dinamico è poi utilizzato dal ProjectionEngineService per modulare le proiezioni dei singoli giocatori.

**4.2. Fonte Dati per il Tiering**

* Tabella team\_historical\_standings: Contiene i piazzamenti, punti, GF, GS delle squadre nelle stagioni precedenti (Serie A e, opzionalmente, Serie B).
* File di configurazione config/team\_tiering\_settings.php: Definisce tutti i parametri per il calcolo.

**4.3. Logica di Calcolo del Punteggio Forza e Assegnazione Tier (gestita da TeamTieringService)**

1. **Selezione Squadre Attive:** Il servizio considera le squadre marcate come attive per la lega e la stagione target (es. serie\_a\_team = true per la Serie A).
2. **Lookback Storico:** Per ogni squadra attiva, vengono recuperati i dati da team\_historical\_standings per un numero configurabile di stagioni precedenti (lookback\_seasons\_for\_tiering).
3. **Calcolo Punteggio Stagione Individuale:**
   * Per ogni stagione storica recuperata, si calcola un "punteggio stagione grezzo" basato su metriche definite (metric\_weights in config), come punti, differenza reti, gol fatti, e posizione in classifica (invertita).
   * **Moltiplicatore di Lega:** Il punteggio stagione grezzo viene moltiplicato per un fattore che riflette la forza relativa della lega in cui è stato ottenuto (es. performance in Serie B "valgono meno" di quelle in Serie A, definito in league\_strength\_multipliers).
4. **Calcolo Punteggio Forza Complessivo:**
   * I punteggi stagione (aggiustati per lega) vengono combinati in un punteggio forza complessivo tramite una media pesata, dove le stagioni più recenti hanno un peso maggiore (season\_weights in config).
   * **Gestione Neopromosse/Dati Mancanti:** Se una squadra ha dati storici insufficienti (o nulli) nel periodo di lookback, le viene assegnato un punteggio grezzo di default (newly\_promoted\_raw\_score\_target), pensato per collocarla in un tier di partenza predefinito (newly\_promoted\_tier\_default).
5. **Normalizzazione Punteggi (Opzionale):**
   * Se configurato (normalization\_method = 'min\_max'), i punteggi forza grezzi di tutte le squadre attive vengono scalati in un range comune (es. 0-100) per facilitare l'applicazione di soglie assolute.
6. **Assegnazione Tier:**
   * Il tier finale (da 1 a 5) viene assegnato in base al punteggio (normalizzato o grezzo) confrontandolo con:
     + Soglie fisse predefinite (tier\_thresholds\_source = 'config', valori in tier\_thresholds\_config).
     + Oppure, soglie calcolate dinamicamente basate su percentili dei punteggi di tutte le squadre attive (tier\_thresholds\_source = 'dynamic\_percentiles', valori in tier\_percentiles\_config).
7. **Aggiornamento Database:** Il campo tier nella tabella teams viene aggiornato con il nuovo valore calcolato.

4.4. Configurazione (config/team\_tiering\_settings.php)

Questo file definisce tutti i parametri per il TeamTieringService, inclusi:

* lookback\_seasons\_for\_tiering, season\_weights, metric\_weights.
* league\_strength\_multipliers (es. ['Serie A' => 1.0, 'Serie B' => 0.7]).
* normalization\_method.
* tier\_thresholds\_source, tier\_thresholds\_config, tier\_percentiles\_config.
* newly\_promoted\_tier\_default, newly\_promoted\_raw\_score\_target.
* Configurazioni API endpoint e ID competizioni (es. serie\_a\_competition\_id, serie\_b\_competition\_id).

4.5. Gestione Squadre Attive per Lega

Il comando Artisan teams:set-active-league viene utilizzato per impostare quali squadre sono considerate partecipanti a una specifica lega (es. Serie A) per una stagione target. Questo aggiorna il flag serie\_a\_team nella tabella teams. Questa operazione è preliminare al calcolo dei tier, per assicurare che la normalizzazione e l'assegnazione dei tier avvengano solo sul corretto pool di squadre.

**5. Modulo di Proiezione Performance Calciatori**

**5.1. Dati di Input per le Proiezioni**

* Medie e FantaMedie storiche ponderate (da HistoricalPlayerStat).
* Età del giocatore (da Player->date\_of\_birth).
* Ruolo ufficiale Fantacalcio (da Player->role).
* **Forza/fascia della squadra di appartenenza (Team->tier): CALCOLATO DINAMICAMENTE** dal TeamTieringService.
* Minutaggio/Presenze attese (stima interna basata su storico, età e tier squadra).
* (Futuro) Ruolo tattico reale, status di rigorista/specialista.

**5.2. Logica di Calcolo delle Proiezioni (Stato Attuale e Sviluppi) - Gestita da ProjectionEngineService**

* **Recupero Dati Storici e Ponderazione Stagionale:** Utilizza dati da historical\_player\_stats, con pesi configurabili in config/projection\_settings.php (es. lookback\_seasons, season\_decay\_factor).
* **Aggiustamento per Età:** Applica modificatori basati su età e ruolo, definiti in config/player\_age\_curves.php.
* **Aggiustamento Statistiche per Contesto Squadra (Tier):** Utilizza il Team->tier (calcolato dinamicamente) e i moltiplicatori da config/projection\_settings.php (team\_tier\_multipliers\_offensive, team\_tier\_multipliers\_defensive) per modulare statistiche come gol, assist, gol subiti.
* **Logica Rigoristi:** Identifica probabili rigoristi e proietta i loro rigori calciati/segnati basandosi su parametri in config/projection\_settings.php.
* **Stima Presenze Attese:** Influenzata da storico, età, e tier squadra (team\_tier\_presence\_factor da config).
* **Proiezione Clean Sheet (P/D): AFFINATA.** La probabilità di clean sheet (clean\_sheet\_per\_game\_proj) è calcolata considerando il tier squadra e l'età del giocatore. Il ProjectionEngineService calcola poi il *contributo medio atteso* del bonus clean sheet ( probabilità\_CS \* bonus\_CS\_da\_regole\_lega) e lo aggiunge alla FantaMedia base.
* **Calcolo FantaMedia Proiettata per Partita:** Il FantasyPointCalculatorService calcola una FantaMedia base (senza il bonus CS diretto). Il ProjectionEngineService aggiunge il contributo medio del clean sheet.
* **Calcolo Totali Stagionali Proiettati:** Moltiplica le medie per partita per le presenze proiettate.

**5.3. Output delle Proiezioni**

* FantaMedia Proiettata per Partita (fanta\_media\_proj\_per\_game).
* Fantapunti Totali Stagionali Proiettati (total\_fantasy\_points\_proj).
* Media Voto Proiettata per Partita (mv\_proj\_per\_game).
* Presenze Proiettate (presenze\_proj).
* Proiezioni delle singole statistiche totali per la stagione (gol, assist, etc. in seasonal\_totals\_proj).
* Dettaglio delle statistiche per partita usate per il calcolo della FantaMedia base e il contributo medio del clean sheet aggiunto (stats\_per\_game\_for\_fm\_calc, che ora include avg\_cs\_bonus\_added e clean\_sheet\_probability\_used).

6. Modulo di Valutazione e Identificazione Talenti (Futuro)

(Come da versione 1.2.1)

7. Modulo Strategia d'Asta (Futuro)

(Come da versione 1.2.1)

**8. Struttura Applicativa Laravel (Alto Livello)**

**8.1. Modelli Principali (Eloquent)**

* App\Models\Player
* App\Models\Team
* App\Models\HistoricalPlayerStat
* App\Models\UserLeagueProfile
* App\Models\ImportLog
* **App\Models\TeamHistoricalStanding (NUOVO)**
* (Futuri) App\Models\PlayerTacticalNote, App\Models\AuctionPlan, App\Models\AuctionPlanTarget.

**8.2. Servizi Chiave**

* Logica di Importazione (in App\Imports\ e Controllers RosterImportController, HistoricalStatsImportController)
* App\Services\DataEnrichmentService: Arricchimento dati giocatori da API.
* **App\Services\TeamDataService (NUOVO):** Recupero dati squadre e classifiche storiche da API.
* App\Services\ProjectionEngineService: Logica di proiezione performance giocatori, ora utilizza tier dinamici e ha gestione affinata del clean sheet.
* App\Services\FantasyPointCalculatorService: Calcolo FantaMedia per singola partita.
* **App\Services\TeamTieringService (NUOVO):** Calcola dinamicamente il tier delle squadre basandosi su dati storici e configurazioni.
* (Futuri) AuctionValueCalculatorService, PlayerTieringService, etc.

**8.3. Controller e Viste Principali**

* App\Http\Controllers\RosterImportController.php
* App\Http\Controllers\HistoricalStatsImportController.php
* App\Http\Controllers\UserLeagueProfileController.php
* Viste Blade per upload e profilo lega (resources/views/uploads/roster.blade.php, historical\_stats.blade.php, league/profile\_edit.blade.php, layouts/app.blade.php).

**8.4. Processi in Background (Jobs)**

* Consigliato per Futuro: Convertire importazioni, arricchimento dati e calcolo tier/proiezioni in Job Laravel per migliorare la responsività e gestire meglio processi lunghi.

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

Questo flusso descrive l'ordine consigliato per preparare i dati e calcolare i tier per una nuova stagione di proiezione.

1. **teams:set-active-league {--target-season-start-year=} {--league-code=SA} {--set-inactive-first=true}**
   * **Scopo:** Definisce quali squadre partecipano a una lega (es. Serie A) per una stagione target. Aggiorna il flag teams.serie\_a\_team. Recupera la lista dei team partecipanti dall'API.
   * **Utilizzo:** Eseguire all'inizio della preparazione di una nuova stagione di proiezione.
   * **Opzioni Chiave:**
     + --target-season-start-year: (Obbligatorio) Anno di inizio della stagione per cui definire le squadre attive (es. 2024 per la stagione 2024-25).
     + --league-code: (Default: SA) Codice della lega.
     + --set-inactive-first: (Default: true) Se impostato, prima disattiva (serie\_a\_team=false) tutte le squadre, poi attiva solo quelle ricevute dall'API per la lega e stagione specificate.
   * **Esempio:** php artisan teams:set-active-league --target-season-start-year=2024 --league-code=SA
   * *File:* app/Console/Commands/TeamsSetActiveLeague.php
2. **teams:map-api-ids {--season=} {--competition=SA}**
   * **Scopo:** Associa i team locali con i loro ID API da Football-Data.org. Popola teams.api\_football\_data\_team\_id.
   * **Utilizzo:** Eseguire dopo aver popolato la tabella teams (tramite seeder o import CSV) e dopo teams:set-active-league se si vogliono mappare specificamente i team attivi in una lega. Utile per assicurare che i team creati manualmente (es. da CSV) vengano arricchiti con l'ID API se disponibili.
   * **Opzioni Chiave:**
     + --competition: (Default: SA) Codice della competizione per cui mappare gli ID.
     + --season: (Opzionale) Può influenzare quali team l'API restituisce, a seconda dell'endpoint.
   * **Esempio:** php artisan teams:map-api-ids --competition=SA
   * *File:* app/Console/Commands/MapTeamApiIdsCommand.php
3. **teams:import-standings-file {filepath} {--season-start-year=} {--league-name="Serie A"} {--create-missing-teams=false} {--default-tier-for-new=4} {--is-serie-a-league=true}**
   * **Scopo:** Importa classifiche storiche da file CSV. Può creare team mancanti nella tabella teams.
   * **Utilizzo:** Per popolare lo storico classifiche, specialmente per stagioni/leghe non coperte da API o per bootstrap iniziale.
   * **Opzioni Chiave:** Vedi documentazione precedente per dettagli completi.
   * **Esempio:** php artisan teams:import-standings-file storage/app/import/classifica\_serie\_a\_2021-22.csv --season-start-year=2021 --create-missing-teams=true
   * *File:* app/Console/Commands/TeamsImportStandingsFile.php
4. **teams:fetch-historical-standings {--season=} {--all-recent=} {--competition=SA}**
   * **Scopo:** Recupera classifiche storiche da API Football-Data.org e le salva in team\_historical\_standings. Può aggiornare teams.api\_football\_data\_team\_id se trova un team per nome.
   * **Utilizzo:** Per popolare/aggiornare automaticamente lo storico classifiche per le stagioni/leghe accessibili via API.
   * **Opzioni Chiave:** Vedi documentazione precedente.
   * **Esempio:** php artisan teams:fetch-historical-standings --all-recent=3 --competition=SA
   * *File:* app/Console/Commands/TeamsFetchHistoricalStandings.php
5. **teams:update-tiers {targetSeasonYear}**
   * **Scopo:** Esegue il TeamTieringService per ricalcolare e aggiornare i tier delle squadre (marcate come attive per la lega target, es. serie\_a\_team=true) per la targetSeasonYear specificata (es. "2024-25").
   * **Utilizzo:** Eseguire dopo aver aggiornato i dati storici delle classifiche (passi 3 e 4) e dopo aver definito le squadre attive per la stagione target (passo 1, se si usa teams:set-active-league). Questo prepara i tier corretti che verranno usati dal ProjectionEngineService.
   * **Argomenti:**
     + targetSeasonYear: (Obbligatorio) La stagione PER CUI calcolare i tier, formato "YYYY-YY" (es. "2024-25").
   * **Esempio:** php artisan teams:update-tiers 2024-25
   * *File:* app/Console/Commands/TeamsUpdateTiers.php
6. **players:enrich-data {--player\_id=all} {--player\_name=} {--delay=SECONDS}**
   * **Scopo:** Arricchisce dati giocatori (data di nascita, etc.) da API Football-Data.org.
   * **Utilizzo:** Dopo l'importazione del roster e periodicamente.
   * *File:* app/Console/Commands/EnrichPlayerDataCommand.php
7. **test:projection {playerId}**
   * **Scopo:** Testa ProjectionEngineService per un giocatore, usando i tier e i dati più recenti.
   * **Utilizzo:** Per debug e analisi singola.
   * *File:* app/Console/Commands/TestPlayerProjectionCommand.php

**Workflow Consigliato per Inizio Nuova Stagione di Proiezione (es. 2025-26):**

1. **Aggiorna Dati Storici Stagione Conclusa (2024-25):**
   * php artisan teams:fetch-historical-standings --season=2024 --competition=SA (per Serie A 24-25 via API)
   * php artisan teams:import-standings-file ... --season-start-year=2024 --league-name="Serie B" ... (per Serie B 24-25 via CSV, se necessario)
2. **Definisci Squadre Attive per Nuova Stagione (2025-26):**
   * php artisan teams:set-active-league --target-season-start-year=2025 --league-code=SA --set-inactive-first=true
3. **Aggiorna/Mappa ID API (Opzionale se set-active-league o fetch-historical-standings li gestiscono):**
   * php artisan teams:map-api-ids --competition=SA
4. **Calcola i Nuovi Tier per la Stagione Target (2025-26):**
   * php artisan teams:update-tiers 2025-26
   * (Itera con modifiche a config/team\_tiering\_settings.php e riesegui se necessario per calibrare i tier)
5. **Importa Nuovo Roster Giocatori (2025-26):** Tramite UI.
6. **Arricchisci Dati Nuovi Giocatori:**
   * php artisan players:enrich-data
7. **Genera Proiezioni:** Ora il ProjectionEngineService userà i tier aggiornati.

**9. Considerazioni Aggiuntive e Sviluppi Futuri**

* **Qualità dei Dati:** Fondamentale per l'accuratezza del tiering e delle proiezioni. La completezza dello storico classifiche (incluse leghe inferiori per neopromosse) migliora il tiering.
* **Manutenzione:** Aggiornamento annuale del flag serie\_a\_team tramite teams:set-active-league, importazione/fetch classifiche stagionali, potenziale ricalibrazione dei parametri di tiering e proiezione.
* **Calibrazione Modelli:** I servizi (TeamTieringService, ProjectionEngineService) e le loro configurazioni (config/team\_tiering\_settings.php, config/projection\_settings.php, config/player\_age\_curves.php) sono potenti ma richiedono un processo iterativo di test e "tuning" per ottimizzare i risultati in base alla propria sensibilità ed esperienza.
* **Usabilità (UX/UI):** Futuri sviluppi potrebbero includere interfacce per visualizzare e modificare i tier, e per lanciare alcuni di questi processi di aggiornamento dati.
* **Gestione Rate Limit API:** Assicurarsi che i comandi che usano API (specialmente se eseguiti in batch per più stagioni/leghe) includano pause adeguate (es. sleep()) per rispettare i limiti dell'API.
* **Machine Learning:** Possibile potenziamento futuro.
* **Supporto Modalità Mantra.**

Questo documento dovrebbe coprire in modo esauriente lo stato attuale del progetto e le funzionalità implementate.