**Consolidated Technical Document: FantaProject - Applicazione Laravel per Proiezioni e Strategia Asta Fantacalcio**

Versione Consolidata: 1.3 (basata sull'integrazione di tutte le versioni fornite fino al 30 maggio 2025)

Data Consolidamento: 30 maggio 2025

**Storico Documenti Sorgente Integrati:**

* Fanta-asta v1.0.docx (Riferimento base)
* Fanta-asta v1.1.docx (Aggiornamenti a Database, Servizi, Modulo Proiezioni)
* Fanta-asta v1.2.docx (Integrazione API, arricchimento dati, comando Artisan)
* Fanta-asta v1.2.1.docx (Dettagli implementativi API, ProjectionEngineService)
* Fanta-asta v1.2.2.docx (Tiering Dinamico, Nuovi Comandi Artisan, Affinamenti Proiezioni - Concettualmente v1.3)
* command.docx (Dettaglio Comandi Artisan Iniziali e Modifiche)
* Configurazione modulazione.docx (Dettaglio config/player\_age\_curves.php e config/projection\_settings.php)

**Indice:**

1. Obiettivo del Progetto
2. Architettura di Sistema e Tecnologie
3. Gestione dei Dati 3.1. Fonti Dati Primarie (Input Utente) 3.2. Fonti Dati per Arricchimento (API Esterne e Dati Qualitativi) 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. 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 5.3. Output delle Proiezioni
6. File di Configurazione Chiave 6.1. config/player\_age\_curves.php 6.2. config/projection\_settings.php 6.3. config/team\_tiering\_settings.php
7. Modulo di Valutazione e Identificazione Talenti (Futuro) 7.1. Calcolo del "Valore d'Asta Interno" 7.2. Identificazione Giocatori Sottovalutati ("Scommesse")
8. Modulo Strategia d'Asta (Futuro) 8.1. Configurazione Lega Fantacalcistica Utente 8.2. Suddivisione Giocatori in Fasce (Tiering) 8.3. Pianificazione Budget per Reparto 8.4. Gestione "Coppie" Titolare/Riserva 8.5. Gestione Diversificazione/Concentrazione per Squadra 8.6. Generazione Lista d'Asta Finale
9. Struttura Applicativa Laravel (Alto Livello) 9.1. Modelli Principali (Eloquent) 9.2. Servizi Chiave 9.3. Controller e Viste Principali 9.4. Processi in Background (Jobs) 9.5. Comandi Artisan Personalizzati (Flusso Operativo e Dettagli)
10. 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 (es. MySQL, PostgreSQL), configurato in config/database.php.
* **Frontend:** Blade templates (files in resources/views/ ). JavaScript con resources/js/app.js e bootstrap.js (come da webpack.mix.js). Possibile utilizzo di JavaScript (es. Livewire, Vue.js o Alpine.js) per interattività.
* **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. Fonti Dati Primarie (Input Utente)**

* **File XLSX Roster Ufficiale:** L'applicazione permette l'upload di file XLSX (es. da Fantacalcio.it) tramite interfaccia web (gestito da RosterImportController ).
  + Contenuto: Lista ufficiale dei calciatori per la stagione, ruoli ufficiali (P, D, C, A) e ruoli Mantra (Rm) secondo la piattaforma Fantacalcio, quotazioni iniziali di riferimento (CRD), ID piattaforma Fantacalcio (fanta\_platform\_id).
  + Un tag/titolo viene estratto dalla prima riga del foglio "Tutti" e salvato in ImportLog.
  + Nota: Le CRD ufficiali sono un valore di riferimento/benchmark, non la base d'asta (che parte da 1 credito per ogni giocatore).
* **File XLSX Statistiche Storiche Giocatori:** L'applicazione permette l'upload di file XLSX contenenti le statistiche dei giocatori delle stagioni precedenti (formato atteso: Riga 1 come titolo/tag, Riga 2 con intestazioni Id, R, Rm, Nome, Squadra, Pv, Mv, Fm, Gf, Gs, Rp, Rc, R+, R-, Ass, Amm, Esp, Au ). Il caricamento avviene tramite interfaccia web (gestito da HistoricalStatsImportController ).
  + La stagione viene derivata dal nome del file. L'ID giocatore nel file viene usato come player\_fanta\_platform\_id.
* **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.
  + Permette di creare automaticamente nella tabella teams le squadre presenti nel CSV ma non nel database.

#### **3.2. Fonti Dati per Arricchimento (API Esterne e Dati Qualitativi)**

* **API Utilizzata:** Football-Data.org (v4).
  + Configurazione API: Chiave API memorizzata in .env (FOOTBALL\_DATA\_API\_KEY) e acceduta tramite config/services.php. URI Base memorizzato in .env (FOOTBALL\_DATA\_API\_BASE\_URI) e acceduto tramite config/services.php.
* **Dati Giocatore Recuperati (tramite DataEnrichmentService):**
  + Data di nascita del giocatore (date\_of\_birth), utilizzata per calcolare l'età nelle proiezioni.
  + ID del giocatore sull'API esterna (api\_football\_data\_id), memorizzato per ottimizzare chiamate future.
  + Posizione/ruolo dettagliato fornito dall'API (detailed\_position), memorizzato per futuri affinamenti tattici.
* **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 Qualitativi Curati (Manualmente o Futuri Servizi):**
  + Probabili rigoristi.
  + Giocatori che ricoprono ruoli tattici diversi da quelli ufficiali (es. difensori offensivi), parzialmente coperto da detailed\_position.
  + Informazioni su gerarchie (titolari/riserve) per identificare "coppie".
* **Fonti Possibili per Statistiche Avanzate (Sviluppo Futuro):** API pubbliche/freemium (es. football-data.org), siti di statistiche (es. FBref, WhoScored tramite scraping etico e conforme ai ToS), dati StatsBomb Open Data. Statistiche come xG, xA, SCA.
* **Dati Potenziali Futuri dall'API:** Nazionalità giocatori, piede preferito, altezza/peso.

#### **3.3. Database Applicativo**

Il database memorizza i seguenti dati principali attraverso i modelli Eloquent:

* **players (App\Models\Player):**
  + Anagrafica base e dati Fantacalcio: fanta\_platform\_id, name, team\_name (nome squadra da roster), team\_id (FK a teams ), role (Classic: P,D,C,A), initial\_quotation, current\_quotation, fvm.
  + Campi arricchiti da API: api\_football\_data\_id (integer, unique), date\_of\_birth (date), detailed\_position (string).
  + Supporta SoftDeletes.
* **teams (App\Models\Team):**
  + name (nome ufficiale), short\_name (opzionale).
  + api\_football\_data\_team\_id (integer, unique, per mapping API).
  + serie\_a\_team (boolean), per indicare partecipazione alla Serie A nella stagione target, gestito da teams:set-active-league.
  + tier (integer), rappresenta una proiezione della forza della squadra per la stagione corrente, inizialmente da TeamSeeder, poi calcolato dinamicamente da TeamTieringService.
  + Opzionale: logo\_url.
* **historical\_player\_stats (App\Models\HistoricalPlayerStat):** (Precedentemente HistoricalSeasonStat )
  + Collega un player\_fanta\_platform\_id a statistiche per una season\_year.
  + Include team\_id (FK, squadra di quella stagione), team\_name\_for\_season.
  + role\_for\_season (Classic), mantra\_role\_for\_season (memorizzato come stringa o JSON array).
  + Metriche: Pv, Mv, Fm, Gf, Gs, Rp, Rc, R+, R-, Ass, Amm, Esp, Au.
* **user\_league\_profiles (App\Models\UserLeagueProfile):**
  + Memorizza le configurazioni della lega dell'utente, inclusi nome lega, budget totale, numero di giocatori per ruolo, numero partecipanti, e scoring\_rules (JSON).
* **import\_logs (App\Models\ImportLog):**
  + Traccia le operazioni di importazione file (roster, storico). Include nome file, tipo, stato, dettagli, conteggi righe.
* **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).
  + Colonne previste: team\_id (FK), api\_football\_data\_team\_id (opzionale), season\_year, league\_name (es. "Serie A"), position, played\_games, won, draw, lost, points, goals\_for, goals\_against, goal\_difference.
* **player\_tactical\_notes (App\Models\PlayerTacticalNote - Futuro):**
  + Attributi speciali: rigorista, ruolo tattico offensivo/difensivo, specialista calci piazzati. Collegata a Players.
* **auction\_plans (App\Models\AuctionPlan) e auction\_plan\_targets (App\Models\AuctionPlanTarget - Futuro):**
  + Piani d'asta dell'utente, budget per reparto, giocatori target con bid personalizzati.

### **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)**

* **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).
* **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).
* **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 (league\_strength\_multipliers in config).
* **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).
* **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).
* **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).
* **Aggiornamento Database:** Il campo tier nella tabella teams viene aggiornato con il nuovo valore calcolato.

#### **4.4. 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.

### **5. Modulo di Proiezione Performance Calciatori**

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

* Medie e FantaMedie storiche ponderate (da HistoricalPlayerStats).
* Età del giocatore (calcolata da date\_of\_birth recuperata via API e memorizzata in Players).
* Ruolo ufficiale Fantacalcio (da Players.role).
* Forza/fascia della squadra di appartenenza (Team.tier): Inizialmente da seeder, poi CALCOLATO DINAMICAMENTE dal TeamTieringService.
* Minutaggio/Presenze attese (stima interna basata su storico, età e tier squadra).
* Status di rigorista/specialista calci piazzati (da PlayerTacticalNotes - sviluppo futuro).
* Ruolo tattico reale (se diverso e identificato, da PlayerTacticalNotes o dedotto da detailed\_position API - sviluppo futuro).

#### **5.2. Logica di Calcolo delle Proiezioni (gestita da ProjectionEngineService)**

* **Recupero Dati Storici:** Caricamento delle statistiche delle ultime N stagioni (default 3 o configurabile in config/projection\_settings.php ) per il giocatore da HistoricalPlayerStats.
  + Se non presenti, si utilizzano proiezioni di default basate su ruolo, tier squadra (attuale) ed età del giocatore (logica in getDefaultStatsPerGameForRole() e estimateDefaultPresences() nel ProjectionEngineService).
* **Ponderazione Stagionale:** Pesi decrescenti per le stagioni più vecchie (es. 50%-33%-17% per 3 stagioni, o basato su season\_decay\_factor ). La logica può usare un sistema di pesi decrescenti (N, N-1, ..., 1) normalizzati.
* **Calcolo Medie Ponderate per Partita:** Per MV, gol/partita, assist/partita, cartellini/partita, ecc..
* **Aggiustamento per Età ("Curva di Rendimento"):** (Implementato)
  + Viene calcolato un ageModifier basato sull'età del giocatore (da date\_of\_birth) e su curve di rendimento definite per ruolo in config/player\_age\_curves.php (fasi di crescita, picco, declino).
  + Questo modificatore influenza: Media Voto (con effetto smorzato mv\_effect\_ratio ), gol/partita, assist/partita, probabilità di clean sheet (leggermente, cs\_age\_effect\_ratio ) e presenze attese (presenze\_growth\_effect\_ratio, presenze\_decline\_effect\_ratio ).
* **Aggiustamento Statistiche per Contesto Squadra (Tier):** (Implementato)
  + Le squadre di Serie A sono classificate in fasce (Tier 1: Top, Tier 2: Europa, etc.).
  + Utilizza il Team->tier (calcolato dinamicamente ) e i moltiplicatori da config/projection\_settings.php (team\_tier\_multipliers\_offensive, team\_tier\_multipliers\_defensive).
  + Le statistiche offensive (gol, assist) per partita sono moltiplicate per un tierMultiplierOffensive.
  + Le statistiche difensive (gol subiti per portieri/difensori) sono modulate da un tierMultiplierDefensive (inverso).
* **Stima Presenze Attese:** (Implementato)
  + Parte dalla media ponderata delle partite giocate storicamente (avg\_games\_played).
  + Modulata da un presenzeTierFactor (derivato dal team\_tier\_presence\_factor in config ) e da un presenzeAgeFactor (derivato dall'ageModifier ).
  + Limitata a un range realistico (es. min 5, max 38 partite, configurabile).
* **Logica Rigoristi:** (Implementata)
  + Identifica probabili rigoristi e proietta i loro rigori calciati/segnati basandosi su parametri in config/projection\_settings.php (es. penalty\_taker\_lookback\_seasons, min\_penalties\_taken\_threshold, league\_avg\_penalties\_awarded, penalty\_taker\_share, default\_penalty\_conversion\_rate).
* **Proiezione Clean Sheet per Partita (Difensori/Portieri):** (Affinata)
  + La probabilità di clean sheet (clean\_sheet\_per\_game\_proj) è calcolata considerando il tier squadra (da clean\_sheet\_probabilities\_by\_tier in config ) e l'età del giocatore (cs\_age\_effect\_ratio in player\_age\_curves.php).
  + 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:** (Implementato)
  + Le statistiche medie per partita aggiustate vengono passate al FantasyPointCalculatorService.
  + Il servizio calcola la FantaMedia per partita attesa (base\_fanta\_media\_per\_game\_proj), basata sulle scoring\_rules della lega dell'utente.
  + Il ProjectionEngineService aggiunge il contributo medio del clean sheet alla FantaMedia base.
* **Calcolo Totali Stagionali Proiettati:** (Implementato)
  + FantaMedia per partita \* presenze attese = Fantapunti totali stagionali.
  + Singole stats medie per partita \* presenze attese = Totali stagionali per statistica.
* **Sviluppi Futuri Identificati per la Logica di Proiezione:**
  + Affinamento parametri peakAgeConfig per l'età (ora player\_age\_curves.php).
  + Affinamento proiezione clean sheet (più data-driven).
  + Tiering Dinamico Squadre (implementato, ma calibrazione continua).
  + Aggiustamento per Ruolo Tattico Specifico (basato su detailed\_position API e futuro PlayerTacticalNote).
  + Considerazione della regressione verso la media.

#### **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, ecc. in seasonal\_totals\_proj).
* Statistiche medie per partita usate per il calcolo della FantaMedia (stats\_per\_game\_for\_fm\_calc), che include il contributo medio del clean sheet aggiunto e la probabilità CS utilizzata (avg\_cs\_bonus\_added, clean\_sheet\_probability\_used).
* Un "Breakout Score" o indicatore di potenziale di crescita (opzionale, futuro).

### **6. File di Configurazione Chiave**

#### **6.1. config/player\_age\_curves.php**

* **Scopo Generale:** Definisce come l'età di un giocatore influenza le proiezioni delle sue performance. Specifica le diverse fasi della carriera (sviluppo, picco, mantenimento, declino) per differenti ruoli e i modificatori quantitativi.
* **Utilizzo Principale:** Usato dal ProjectionEngineService per calcolare un ageModifier generale e applicarlo a diverse statistiche proiettate (MV, gol, assist, CS, presenze).
* **Struttura Dettagliata:**
  + descrizione: Scopo del file.
  + disclaimer: Avviso sulla generalizzazione delle curve.
  + dati\_ruoli: Array associativo per raggruppamento di ruolo (es. 'P', 'D\_CENTRALE', 'C', 'A').
    - fasi\_carriera: Età di transizione (sviluppo\_fino\_a, picco\_inizio, picco\_fine, mantenimento\_fino\_a (opzionale), declino\_da).
    - note\_picco\_declino: Note testuali.
    - growth\_factor: Incremento % annuo in fase di sviluppo.
    - decline\_factor: Decremento % annuo in fase di declino.
    - young\_cap: Limite superiore modificatore età per giovani.
    - old\_cap: Limite inferiore modificatore età per anziani.
    - age\_modifier\_params: Come l'ageModifier si applica a diverse statistiche.
      * mv\_effect\_ratio: Frazione dell'effetto età sulla Media Voto.
      * cs\_age\_effect\_ratio: Frazione dell'effetto età sulla probabilità di Clean Sheet.
      * presenze\_growth\_effect\_ratio: Frazione dell'effetto positivo età sulle presenze.
      * presenze\_decline\_effect\_ratio: Frazione/amplificazione dell'effetto negativo età sulle presenze.
      * presenze\_growth\_cap: Limite massimo incremento presenze per giovani.
      * presenze\_decline\_cap: Limite minimo riduzione presenze per anziani.
* **Esempi di Configurazione:**
  + Attaccanti (A) con picco breve e intenso: sviluppo\_fino\_a: 22, picco\_inizio: 23, picco\_fine: 28, declino\_da: 29, growth\_factor: 0.030, decline\_factor: 0.040.
  + Portieri (P) con declino più lento: decline\_factor: 0.010, old\_cap: 0.85.
  + Impatto età su MV e presenze: mv\_effect\_ratio: 0.6, presenze\_growth\_effect\_ratio: 0.3. Un ageModifier di 1.10 (giocatore giovane) con mv\_effect\_ratio: 0.6 porta a un moltiplicatore MV di 1 + (0.10 \* 0.6) = 1.06. Con presenze\_growth\_effect\_ratio: 0.3, il moltiplicatore presenze sarebbe 1 + (0.10 \* 0.3) = 1.03.

#### **6.2. config/projection\_settings.php**

* **Scopo Generale:** Contiene parametri che governano il ProjectionEngineService. Definisce recupero e peso dati storici, aggiustamenti (diversi dall'età), gestione rigoristi, default, fallback, output.
* **Utilizzo Principale:** Cuore della configurazione del ProjectionEngineService, permette calibrazione fine del modello predittivo.
* **Struttura Dettagliata (con Esempi):**
  + **Parametri Rigoristi:**
    - penalty\_taker\_lookback\_seasons: (es. 2) Stagioni per identificare rigorista.
    - min\_penalties\_taken\_threshold: (es. 3) Min rigori calciati per essere potenziale rigorista.
    - league\_avg\_penalties\_awarded: (es. 0.20) Media rigori assegnati per squadra a partita.
    - penalty\_taker\_share: (es. 0.85) Quota rigori calciati dal rigorista designato.
    - default\_penalty\_conversion\_rate: (es. 0.75) Tasso conversione rigori di fallback.
    - min\_penalties\_taken\_for\_reliable\_conversion\_rate: (es. 5) Min rigori calciati per tasso conversione personale affidabile.
  + **Gestione Dati Storici e Medie:**
    - lookback\_seasons: (es. 4) Stagioni storiche per medie ponderate generali.
    - season\_decay\_factor: (es. 0.75) Fattore decadimento pesi stagioni vecchie. E.g., pesi (1), (1*0.75), (1*0.75\*0.75) normalizzati. (Nota: la logica effettiva potrebbe usare pesi N, N-1..1 ).
    - fields\_to\_project: Array di statistiche da HistoricalPlayerStat da proiettare (es. 'avg\_rating', 'goals\_scored').
    - min\_games\_for\_reliable\_avg\_rating: (es. 10) Min partite per MV stagionale affidabile.
  + **Impatto del Tier Squadra:**
    - default\_team\_tier: (es. 3) Tier di fallback.
    - team\_tier\_multipliers\_offensive: (Tier => Moltiplicatore) per stats offensive. Es: '1' => 1.20 (+20% per Tier 1).
    - team\_tier\_multipliers\_defensive: (Tier => Moltiplicatore) per stats difensive. Es: '1' => 0.80 (-20% gol subiti per Tier 1).
    - team\_tier\_presence\_factor: (Tier => Moltiplicatore) per presenze attese. Es: '5' => 0.95 (-5% presenze per Tier 5).
    - offensive\_stats\_fields: Campi influenzati da team\_tier\_multipliers\_offensive.
    - defensive\_stats\_fields\_goalkeeper: Campi portieri influenzati da team\_tier\_multipliers\_defensive.
  + **Clean Sheet:**
    - league\_average\_clean\_sheet\_rate\_per\_game: (es. 0.28) Tasso medio CS per partita.
    - clean\_sheet\_probabilities\_by\_tier: (Tier => Probabilità CS base). Es: 1 => 0.40, 3 => 0.28, 5 => 0.18.
    - max\_clean\_sheet\_probability: (es. 0.8) Limite massimo probabilità CS proiettata.
  + **Valori di Default e Fallback:**
    - default\_player\_age: (es. 25) Età se date\_of\_birth non disponibile.
    - fallback\_mv\_if\_no\_history: (es. 5.5) MV di default senza storico.
    - fallback\_fm\_if\_no\_history: (es. 5.5) FantaMedia di default senza storico.
    - fallback\_gp\_if\_no\_history: (es. 0) Presenze di default senza storico.
    - min\_projected\_presences / max\_projected\_presences: (es. 5 / 38) Limiti presenze proiettate.
  + **Configurazione Output Proiezioni:**
    - fields\_to\_project\_output: Mappa output finale a come calcolarlo. Es: 'goals\_scored\_proj' => ['type' => 'sum', 'source\_per\_game' => 'goals\_scored', 'default\_value' => 0].
* **Esempi di Configurazione:**
  + Logica Rigoristi: Abbassare league\_avg\_penalties\_awarded se i rigori sono rari. Aumentare penalty\_taker\_share (es. a 0.90) se il rigorista designato calcia quasi tutti i rigori.
  + Ponderazione Storico: Aumentare season\_decay\_factor per dare più peso alle stagioni recenti o usare pesi espliciti.
  + Impatto Tier: Se Tier 1 segna +30%, team\_tier\_multipliers\_offensive: '1' => 1.30. Se Tier 5 subisce +25% gol, team\_tier\_multipliers\_defensive: '5' => 1.25 (interpretato come impatto sui gol subiti dalla squadra, quindi il moltiplicatore per il giocatore potrebbe essere inverso per statistiche positive come parate o positivo per gol subiti individuali).
  + Valori Default Nuovi Giocatori: fallback\_mv\_if\_no\_history, fallback\_fm\_if\_no\_history devono essere stime prudenti ma ragionevoli.

#### **6.3. config/team\_tiering\_settings.php**

* **Scopo:** Definisce tutti i parametri per il TeamTieringService.
* **Contenuti Chiave:**
  + lookback\_seasons\_for\_tiering, season\_weights, metric\_weights (per punti, diff. reti, gol fatti, posizione).
  + league\_strength\_multipliers (es. ['Serie A' => 1.0, 'Serie B' => 0.7]).
  + normalization\_method (es. 'min\_max' o null).
  + tier\_thresholds\_source ('config' o 'dynamic\_percentiles').
  + tier\_thresholds\_config (soglie fisse se tier\_thresholds\_source = 'config').
  + tier\_percentiles\_config (percentili se tier\_thresholds\_source = 'dynamic\_percentiles').
  + newly\_promoted\_tier\_default, newly\_promoted\_raw\_score\_target (per squadre con dati storici insufficienti).
  + Configurazioni API: api\_football\_data.serie\_a\_competition\_id, api\_football\_data.serie\_b\_competition\_id, standings\_endpoint.
  + Cache TTLs (definite anche in cache.php).
  + Delay API (definito anche in services.php come api\_delay\_seconds).

### **7. Modulo di Valutazione e Identificazione Talenti (Futuro)**

[Sezioni 5.1 e 5.2 da Fanta-asta v1.0.docx / v1.1 / v1.2.1 / v1.2.2 / v1.2 ]

#### **7.1. Calcolo del "Valore d'Asta Interno" (Target Price)**

* Basato sulla FantaMedia Proiettata e sulla scarsità del ruolo, utilizzando concetti come VORP (Value Over Replacement Player) o simili, rapportati al budget totale della lega.
* Questo valore rappresenta quanto l'utente dovrebbe essere disposto a pagare per un giocatore partendo da 1 credito.
* È indipendente (ma confrontabile) dalla CRD ufficiale di Fantacalcio.it.

#### **7.2. Identificazione Giocatori Sottovalutati ("Scommesse")**

* Confronto tra il "Valore d'Asta Interno" calcolato e la CRD ufficiale (o la percezione generale del mercato).
* Evidenziazione di giocatori con alto "Valore d'Asta Interno" ma bassa CRD, o con alto potenziale di breakout non ancora riflesso nel prezzo.
* Evidenziazione di giocatori potenzialmente sopravvalutati (CRD alta, Valore Interno basso).

### **8. Modulo Strategia d'Asta (Futuro)**

[Sezioni 6.1 a 6.6 da Fanta-asta v1.0.docx / v1.1 / v1.2.1 / v1.2.2 / v1.2 ]

#### **8.1. Configurazione Lega Fantacalcistica Utente**

* L'utente dovrà poter inserire (tramite UserLeagueProfiles ):
  + Budget totale disponibile per l'asta.
  + Numero di giocatori per ruolo da acquistare (P, D, C, A).
  + Regole di punteggio specifiche della lega (per personalizzare le proiezioni di FantaMedia).
  + Numero di partecipanti alla lega (per calibrare la scarsità).

#### **8.2. Suddivisione Giocatori in Fasce (Tiering)**

* I giocatori verranno classificati in fasce (es. Top Player, Semi-Top, Buoni Titolari, Scommesse, Low-Cost) basandosi sul loro "Valore d'Asta Interno" calcolato e/o sulla FantaMedia Proiettata.

#### **8.3. Pianificazione Budget per Reparto**

* L'utente potrà definire percentuali o importi fissi del budget da allocare per portieri, difensori, centrocampisti e attaccanti.
* L'applicazione aiuterà a bilanciare le scelte con il budget disponibile per reparto.

#### **8.4. Gestione "Coppie" Titolare/Riserva**

* Identificazione e suggerimento di potenziali "coppie" (es. portiere titolare + riserva; giocatore titolare + suo backup diretto).
* Valutazione del costo combinato della coppia vs. i punti "slot" attesi.
* Strategia per risparmiare crediti e assicurare copertura per un ruolo.

#### **8.5. Gestione Diversificazione/Concentrazione per Squadra**

* Monitoraggio del numero di giocatori selezionati per ciascuna squadra di Serie A nel piano d'asta dell'utente.
* Possibilità per l'utente di impostare un limite massimo di giocatori per squadra.
* Avvisi in caso di eccessiva concentrazione su singole squadre, specialmente se non di primissima fascia, per mitigare il rischio "annata no".
* Considerazione dell'"effetto hype" per giocatori di squadre top, che potrebbero costare più del loro valore statistico puro. L'app fornirà note strategiche.

#### **8.6. Generazione Lista d'Asta Finale**

* Output di una lista stampabile/esportabile contenente per ogni giocatore target:
  + Nome, Squadra, Ruolo Fantacalcio.
  + CRD Ufficiale (Fantacalcio.it) – come riferimento.
  + Tuo Valore Obiettivo (Calcolato) – guida per l'asta.
  + Tuo Max Bid Consigliato.
  + Fascia assegnata.
  + Note strategiche (es. "Rigorista", "Rischio turnover", "Scommessa").

### **9. Struttura Applicativa Laravel (Alto Livello)**

#### **9.1. Modelli Principali (Eloquent)**

* App\Models\Player (Descritto in Sezione 3.3)
* App\Models\Team (Descritto in Sezione 3.3)
* App\Models\HistoricalPlayerStat (Descritto in Sezione 3.3, precedentemente HistoricalSeasonStat)
* App\Models\UserLeagueProfile (Descritto in Sezione 3.3)
* App\Models\ImportLog (Descritto in Sezione 3.3)
* App\Models\TeamHistoricalStanding (Descritto in Sezione 3.3)
* (Futuri) App\Models\PlayerTacticalNote, App\Models\AuctionPlan, App\Models\AuctionPlanTarget.

#### **9.2. Servizi Chiave**

* **Logica di Importazione:**
  + App\Imports\MainRosterImport (usa TuttiSheetImport, FirstRowOnlyImport).
  + App\Imports\HistoricalStatsFileImport (usa TuttiHistoricalStatsImport, FirstRowOnlyImport).
  + Logica precedentemente anche in RosterImportController e HistoricalStatsImportController.
* **App\Services\DataEnrichmentService:** (Implementato Parzialmente )
  + Si connette a Football-Data.org API.
  + Recupera date\_of\_birth, detailed\_position, api\_football\_data\_id per i giocatori.
  + Implementa una logica di matching nomi giocatori e squadre con fallback e punteggio.
  + Utilizza la cache di Laravel per le risposte API.
  + Futuro: Recupero dati storici squadre, statistiche avanzate.
* **App\Services\TeamDataService:** (Implementato)
  + Recupero dati squadre e classifiche storiche da API Football-Data.org.
  + Utilizzato per popolare team\_historical\_standings e per mappare api\_football\_data\_team\_id.
* **App\Services\ProjectionEngineService:** (Implementazione Avanzata)
  + Cuore del sistema, implementa la logica di proiezione (storico ponderato, età, tier squadra, rigoristi, clean sheet).
  + Calcola medie per partita e totali stagionali.
  + Utilizza config/player\_age\_curves.php e config/projection\_settings.php.
* **App\Services\FantasyPointCalculatorService:** (Implementato)
  + Converte statistiche (per partita) in FantaMedia (per partita) basata sulle scoring\_rules della lega.
* **App\Services\TeamTieringService:** (Implementato)
  + Calcola dinamicamente il tier delle squadre basandosi su dati storici da team\_historical\_standings e configurazioni in config/team\_tiering\_settings.php.
* **(Futuri)** AuctionValueCalculatorService, PlayerTieringService (per giocatori, diverso da quello squadre), AuctionStrategyBuilderService, PairAnalyzerService, TeamConcentrationService.

#### **9.3. Controller e Viste Principali**

* **Controller Implementati:**
  + App\Http\Controllers\RosterImportController.php.
  + App\Http\Controllers\HistoricalStatsImportController.php.
  + App\Http\Controllers\UserLeagueProfileController.php.
* **Viste Implementate (in resources/views/):**
  + uploads/roster.blade.php.
  + uploads/historical\_stats.blade.php.
  + league/profile\_edit.blade.php.
  + layouts/app.blade.php (layout base).
* **Futuri:** Controller e viste per visualizzazione proiezioni, costruzione piano d'asta, gestione tier, ecc..

#### **9.4. Processi in Background (Jobs)**

* **Consigliato/Da Implementare Come Asincroni:**
  + ImportFantacalcioRosterJob: Attualmente logica sincrona. Da convertire per importazioni grandi.
  + ImportHistoricalStatsJob: Attualmente logica sincrona. Da convertire in Job.
  + EnrichPlayerDataJob: Attualmente la logica è nel comando Artisan players:enrich-data. Per arricchire i dati dei giocatori tramite API in background, gestendo rate limiting e retry.
  + FetchTeamHistoricalStandingsJob: (Futuro) Per recuperare i dati storici delle squadre.
  + RecalculateProjectionsJob: (Futuro) Per aggiornare le proiezioni se i dati di base cambiano (es. tier squadre, dati giocatori, regole lega).
  + L'aggiornamento dei tier tramite TeamsUpdateTiers potrebbe anch'esso diventare un job.

#### **9.5. 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 (es. 2024 per 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}**
   * **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. Da eseguire dopo aver popolato la tabella teams (es. con TeamSeeder.php o import CSV ) e dopo teams:set-active-league se si vogliono mappare specificamente i team attivi. Utile per arricchire team creati manualmente.
   * **Opzioni:**
     + --season=YYYY: (Opzionale) Anno di inizio stagione. Può influenzare quali team l'API restituisce.
     + --competition=CODICE\_LEGA: (Default: SA) Codice della competizione (es. SA per Serie A, SB per Serie B).
   * **Esempi:** 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, modello Team, config/services.php, config/team\_tiering\_settings.php.
3. **teams:import-standings-file {filepath} {--season-start-year=} {--league-name="Nome Lega"} {--create-missing-teams=false} {--default-tier-for-new=4} {--is-serie-a-league=true}**
   * **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 o per un setup iniziale massivo.
   * **Argomenti:**
     + filepath: (Obbligatorio) Percorso al file CSV.
   * **Opzioni:**
     + --season-start-year=YYYY: (Obbligatorio) Anno di inizio della stagione dei dati CSV.
     + --league-name="Nome Lega": (Default: Serie A) Nome della lega per i dati importati.
     + --create-missing-teams=true|false: (Default: false) Se true, crea record in teams se squadra non trovata.
     + --default-tier-for-new=TIER: (Default: 4) Tier da assegnare alle squadre create.
     + --is-serie-a-league=true|false: (Default: true) Imposta il flag serie\_a\_team per le squadre create. Usare false per leghe come Serie B.
   * **Esempi:** php artisan teams:import-standings-file storage/app/import/classifica\_serie\_a\_2021-22.csv --season-start-year=2021 --create-missing-teams=true.
   * **File Coinvolti:** app/Console/Commands/TeamsImportStandingsFile.php, modelli Team, TeamHistoricalStanding. Richiede league/csv.
4. **teams:fetch-historical-standings {--season=} {--all-recent=} {--competition=SA}**
   * **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.
     + --all-recent=N: (Opzionale) Scarica le classifiche per le ultime N stagioni recenti.
     + --competition=CODICE\_LEGA: (Default: SA) Il codice della competizione.
   * **Esempi:** php artisan teams:fetch-historical-standings --all-recent=3 --competition=SA. php artisan teams:fetch-historical-standings --season=2023 --competition=SA.
   * **File Coinvolti:** app/Console/Commands/TeamsFetchHistoricalStandings.php, app/Services/TeamDataService.php, modelli Team, TeamHistoricalStanding, config/services.php, config/team\_tiering\_settings.php, config/cache.php.
5. **teams:update-tiers {targetSeasonYear}**
   * **Scopo:** Esegue il TeamTieringService per ricalcolare e aggiornare i tier delle squadre (marcate come attive, es. serie\_a\_team=true) per la targetSeasonYear specificata (es. "2024-25").
   * **Utilizzo:** Eseguire dopo aver aggiornato i dati storici delle classifiche e definito le squadre attive per la stagione target. Questo prepara i tier corretti per il 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}**
   * **Spiegazione:** Arricchisce i dati dei giocatori presenti nella tabella players (es. data di nascita, posizione dettagliata, ID API) interrogando l'API esterna Football-Data.org.
   * **Utilizzo:** Fondamentale per ottenere dati anagrafici accurati (specialmente l'età). Da eseguire dopo l'importazione iniziale del roster e periodicamente.
   * **Opzioni:**
     + --player\_id=all|ID: (Default: all) Specifica se arricchire tutti i giocatori che necessitano di dati (date\_of\_birth O detailed\_position O api\_football\_data\_id a NULL ) o un giocatore specifico tramite ID database locale.
     + --player\_name=NOME: Arricchisce i giocatori il cui nome contiene la stringa specificata (case-insensitive).
     + --delay=SECONDI: (Default: 6 o 7 ) Numero di secondi di attesa tra le chiamate API, per rispettare i rate limit.
   * **Esempi:** 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.
7. **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 o ne crea uno di default.
   * **Utilizzo:** Utile per il debug della logica di proiezione, per verificare l'impatto di modifiche ai parametri di configurazione su un giocatore campione, o per analizzare rapidamente le aspettative per un singolo calciatore. Usa i tier e i dati più recenti.
   * **Argomenti:**
     + playerId: (Obbligatorio) Il fanta\_platform\_id del giocatore da testare.
   * **Esempio:** php artisan test:projection 2170. php artisan test:projection 4220 (Testa Zambo Anguissa).
   * **File Coinvolti:** app/Console/Commands/TestPlayerProjectionCommand.php, app/Services/ProjectionEngineService.php, app/Services/FantasyPointCalculatorService.php, modelli Player, UserLeagueProfile, Team, HistoricalPlayerStat.

* **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 non gestito dai passi precedenti):**
     + 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. Le proiezioni possono essere testate con test:projection o tramite future interfacce web.

### **10. Considerazioni Aggiuntive e Sviluppi Futuri**

* **Qualità dei Dati:** Fondamentale per l'accuratezza del tiering e delle proiezioni. L'accuratezza dell'arricchimento API (matching nomi) è cruciale. La completezza dello storico classifiche (incluse leghe inferiori per neopromosse) migliora il tiering.
* **Manutenzione:** Aggiornamento annuale rose, dati qualitativi, fasce squadre (tier). Gestione squadre promosse/retrocesse tramite teams:set-active-league. 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) richiedono un processo iterativo di test e "tuning".
* **Gestione Rate Limit API:** Assicurarsi che i comandi che usano API includano pause adeguate (es. sleep(), opzione --delay) per rispettare i limiti. Implementare logiche di retry con backoff esponenziale per errori 429.
* **Usabilità (UX/UI):** Interfaccia intuitiva e facile da usare, specialmente durante la preparazione all'asta. Futuri sviluppi potrebbero includere interfacce per visualizzare e modificare i tier, e per lanciare processi di aggiornamento dati.
* **Proiezione Forza Squadra (Tiering Avanzato):** L'assegnazione del tier alle squadre potrebbe evolvere da un sistema basato su storico e config a un sistema più dinamico che analizza anche altri indicatori. (Nota: Già largamente implementato con TeamTieringService).
* **Proiezione Rigoristi:** Priorità alta per migliorare l'accuratezza.
* **Utilizzo detailed\_position:** Integrare la posizione dettagliata dall'API negli aggiustamenti delle proiezioni.
* **Machine Learning:** In futuro, la logica di proiezione potrebbe essere potenziata con modelli di machine learning.
* **Supporto Modalità Mantra:** Estensione per supportare le specificità del sistema Mantra (la base dati per mantra\_role\_for\_season è già presente ).
* **Community Features:** Possibilità per gli utenti (se l'app fosse multi-utente) di condividere anonimamente dati su prezzi d'asta reali per raffinare le stime di mercato.