✅ CHECKLIST COMPLÈTE - API SPORTS BETTEAM

> Document créé le 2026-01-13

> Objectif: Intégration complète de TheSportsDB pour une API de pronostics multi-sports

---

📊 ÉTAT ACTUEL DU PROJET

✅ Déjà Implémenté

- Infrastructure API de base (Express + TypeScript)
- Authentification JWT complète
- Base de données PostgreSQL avec Prisma
- Modèles de données: `Competition`, `Team`, `Match`, `SyncLog`, `User`, `League`, `LeagueMember`, `Bet`
- Documentation Swagger interactive
- Déploiement automatisé (Railway)
- Seed de données pour démo (Ligue 1, Premier League)

⏳ À Implémenter

- Synchronisation complète avec TheSportsDB
- Endpoints CRUD pour les compétitions, matchs, paris
- Services métier (calcul gains, résolution paris)
- Gestion multi-sports
- Mise à jour automatique des scores
- Notifications temps réel

🏆 SPORTS & COMPÉTITIONS - IDS THESPORTSDB

⚽ FOOTBALL (5 ligues)

Compétition	Pays	ID TheSportsDB	Statut
Ligue 1	France	`4334`	✅ En seed
Premier League	Angleterre	`4328`	✅ En seed
La Liga	Espagne	`4335`	⏳ À ajouter
Bundesliga	Allemagne	`4331`	⏳ À ajouter
Serie A	Italie	`4332`	⏳ À ajouter

🏉 RUGBY (2 compétitions)

Compétition	Zone	ID TheSportsDB	Statut
Top 14	France	`4430`	⏳ À ajouter
United Rugby Championship	International	`4446`	⏳ À ajouter

🎾 TENNIS (2 circuits)

Circuit	Type	ID TheSportsDB	Statut
ATP World Tour	Hommes	`4464`	⏳ À ajouter

Circuit	Type	ID TheSportsDB	Statut
WTA Tour	Femmes	4517	À ajouter

BASKETBALL

Compétition	Pays	ID TheSportsDB	Statut
NBA	USA	`4387`	À ajouter

CYCLISME

Circuit	Type	ID TheSportsDB	Statut
UCI World Tour	International	`4465`	À ajouter

HOCKEY SUR GLACE

Compétition	Pays	ID TheSportsDB	Statut
NHL	USA/Canada	`4380`	À ajouter

FORMULE 1

Compétition	Type	ID TheSportsDB	Statut
Formula 1	Mondial	`4370`	À ajouter

TOTAL: 14 compétitions (2 déjà en seed, 12 à ajouter)

CHECKLIST DÉTAILLÉE PAR CATÉGORIE

1. INFRASTRUCTURE & CONFIGURATION

- [ ] **1.1** Configurer variable d'environnement `THESPORTSDB_API_KEY`
- [ ] **1.2** Créer service HTTP client pour TheSportsDB
  - Rate limiting (30 req/min en gratuit)
  - Gestion des erreurs 429 (Too Many Requests)
  - Retry avec backoff exponentiel
  - Logging des appels API
- [ ] **1.3** Créer système de cache Redis/in-memory
  - Cache des compétitions (24h)
  - Cache des équipes (24h)
  - Cache des matchs à venir (1h)
  - Cache des matchs en cours (2min avec clé premium, sinon 15min)
- [ ] **1.4** Configurer CRON jobs pour synchronisation
  - Sync compétitions: 1x/jour
  - Sync équipes: 1x/jour
  - Sync matchs à venir: toutes les 6h
  - Sync résultats: toutes les 15min (ou 2min avec premium)

2. BASE DE DONNÉES & MODÈLES

2.1 Schéma Prisma - Extensions

- [ ] **2.1.1** Enrichir modèle `Competition`

```
- currentSeason: String?
- seasonStart: DateTime?
- seasonEnd: DateTime?
- badge: String? (URL badge haute qualité)
- banner: String? (URL bannière)
```

- ☐ **2.1.2** Enrichir modèle `Team`

```
- stadium: String?
- stadiumThumb: String?
- website: String?
- description: String? @db.Text
- foundedYear: Int?
- jerseyUrl: String?
- fanart: String?
```

- ☐ **2.1.3** Enrichir modèle `Match`

```
- homeGoalDetails: String? (JSON: scoreurs + minutes)
- awayGoalDetails: String? (JSON: scoreurs + minutes)
- spectators: Int?
- referee: String?
- highlightVideo: String? (URL)
- thumbnail: String? (URL affiche match)
```

- ☐ **2.1.4** Créer modèle `Player`

```
model Player {
  id String @id @default(cuid())
  externalId String @unique
  teamId String
  name String
  position String?
  number Int?
  nationality String?
  photoUrl String?
  birthdate DateTime?
  height String?
  weight String?
  team Team @relation(fields: [teamId], references: [id])
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}
```

- ☐ **2.1.5** Créer modèle `Standing` (classements)

```
model Standing {
  id String @id @default(cuid())
  competitionId String
  teamId String
  season String
  position Int
  played Int
  won Int
  drawn Int
  lost Int
  goalsFor Int
  goalsAgainst Int
  goalDifference Int
  points Int
  form String? (ex: "WWDLW")
  competition Competition @relation(fields: [competitionId], references: [id])
  team Team @relation(fields: [teamId], references: [id])
  updatedAt DateTime @updatedAt
  @@unique([competitionId, teamId, season])
}
```

2.2 Migrations

- ☐ **2.2.1** Créer migration pour nouvelles colonnes Competition/Team/Match
- ☐ **2.2.2** Créer migration pour modèle Player

- ☐ **2.2.3** Créer migration pour modèle Standing
- ☐ **2.2.4** Créer indexes pour performances

```
 @@index([competitionId, season])
@@index([teamId])
@@index([position])
```

🔌 3. INTÉGRATION THESPORTSDB - SERVICES

3.1 Service de Synchronisation Compétitions

- ☐ **3.1.1** `syncCompetition(leagueId: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/lookupleague.php?id={id}`
  - Champs à récupérer: name, sport, country, badge, banner, currentSeason
  - Upsert dans table `Competition`
  - Logger dans `SyncLog`
- ☐ **3.1.2** `syncAllCompetitions()`
  - Boucle sur les 14 IDs de compétitions
  - Respect rate limit (max 30/min)
  - Gestion erreurs par compétition

3.2 Service de Synchronisation Équipes

- ☐ **3.2.1** `syncTeamsByCompetition(leagueId: string, season: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/lookup_all_teams.php?id={id}`
  - Champs: name, shortName, logo, stadium, website, foundedYear, jersey, fanart
  - Upsert dans table `Team`
  - Associer à la compétition
- ☐ **3.2.2** `syncAllTeams()`
  - Pour chaque compétition active
  - Récupérer saison courante
  - Synchroniser toutes les équipes

3.3 Service de Synchronisation Matchs

- ☐ **3.3.1** `syncUpcomingMatches(leagueId: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/eventsnextleague.php?id={id}`
  - Champs: dateEvent, strTime, homeTeam, awayTeam, venue, round
  - Créer matches avec status="upcoming"
  - Mapper externalId équipes vers IDs internes
- ☐ **3.3.2** `syncPastMatches(leagueId: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/eventspastleague.php?id={id}`
  - Récupérer les 15 derniers matchs
  - Champs: intHomeScore, intAwayScore, strStatus, strVideo (highlights)
  - Mettre à jour matches avec status="finished"
- ☐ **3.3.3** `syncMatchDetails(eventId: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/lookupevent.php?id={id}`
  - Détails complets: scoreurs, arbitre, spectateurs, thumbnail
  - Enrichir match existant
- ☐ **3.3.4** `syncLiveScores(leagueId: string)` (Premium uniquement)
  - Endpoint: `https://www.thesportsdb.com/api/v2/json/{premium_key}/livescore.php?l={id}`
  - Mise à jour temps réel (toutes les 2min)
  - Status="live", scores en direct

3.4 Service de Synchronisation Joueurs

- ☐ **3.4.1** `syncPlayersByTeam(teamId: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/lookup_all_players.php?id={id}`
  - Champs: name, position, number, nationality, photo, birthdate, height, weight
  - Insérer dans table `Player`
- ☐ **3.4.2** `syncAllPlayers()`
  - Pour chaque équipe active
  - Respect rate limit

3.5 Service de Classements

- ☐ **3.5.1** `syncStandings(leagueId: string, season: string)`
  - Endpoint: `https://www.thesportsdb.com/api/v1/json/{key}/lookuptable.php?l={id}&s={season}`
  - Champs: position, played, won, drawn, lost, goalsfor, goalsagainst, points
  - Upsert dans table `Standing`
- ☐ **3.5.2** `syncAllStandings()`
  - Pour chaque compétition avec classement (football, rugby, NBA, NHL)

🔌 4. ENDPOINTS API REST

4.1 Compétitions

- ☐ **4.1.1** `GET /api/competitions`
  - Query params: ?sport=football&country=France&isActive=true
  - Retour: liste compétitions avec badges
- ☐ **4.1.2** `GET /api/competitions/:id`
  - Détails compétition
  - Inclure: saison courante, nombre d'équipes, prochain match
- ☐ **4.1.3** `GET /api/competitions/:id/teams`
  - Liste équipes de la compétition
  - Avec logos et infos de base
- ☐ **4.1.4** `GET /api/competitions/:id/standings`
  - Classement actuel
  - Trié par position
- ☐ **4.1.5** `POST /api/competitions/sync` (Admin uniquement)
  - Trigger synchronisation manuelle
  - Body: { leagueId?: string } (si vide, sync all)

4.2 Équipes

- ☐ **4.2.1** `GET /api/teams`
  - Query params: ?competitionId=xxx&search=Arsenal
  - Retour: liste équipes
- ☐ **4.2.2** `GET /api/teams/:id`
  - Détails équipe complète
  - Inclure: stade, joueurs, derniers matchs
- ☐ **4.2.3** `GET /api/teams/:id/players`
  - Effectif complet
  - Trié par numéro de maillot
- ☐ **4.2.4** `GET /api/teams/:id/matches`
  - Query params: ?status=upcoming&limit=5
  - Historique et calendrier

4.3 Matchs

- ☐ **4.3.1** `GET /api/matches`
  - Query params:
    - ?sport=football
    - &competitionId=xxx
    - &status=upcoming|live|finished
    - &startDate=2026-01-13
    - &endDate=2026-01-20
    - &limit=50&offset=0
  - Retour: liste matchs avec équipes et compétitions
- ☐ **4.3.2** `GET /api/matches/:id`
  - Détails complet du match
  - Inclure: compositions, scoreurs, stats, highlights
- ☐ **4.3.3** `GET /api/matches/today`
  - Matchs du jour toutes compétitions
- ☐ **4.3.4** `GET /api/matches/live`
  - Matchs en cours
  - Avec scores temps réel si premium

- [ ] **4.3.5** `POST /api/matches/sync` (Admin)
    - Trigger sync matchs
    - Body: { competitionId?: string, type: 'upcoming'|'past' }

4.4 Joueurs

- [ ] **4.4.1** `GET /api/players`
    - Query params: ?teamId=xxx&position=Forward&search=Messi
- [ ] **4.4.2** `GET /api/players/:id`
    - Profil complet joueur
    - Stats, historique

4.5 Ligues (Déjà défini, à implémenter)

- [ ] **4.5.1** `GET /api/leagues`
    - Ligues de l'utilisateur
- [ ] **4.5.2** `POST /api/leagues`
    - Créer une ligue privée
    - Body: { name, description?, isPrivate }
- [ ] **4.5.3** `GET /api/leagues/:id`
    - Détails ligue + classement membres
- [ ] **4.5.4** `POST /api/leagues/:id/join`
    - Rejoindre avec code invitation
    - Body: { inviteCode }
- [ ] **4.5.5** `GET /api/leagues/:id/members`
    - Liste membres avec points
- [ ] **4.5.6** `DELETE /api/leagues/:id/members/:userId` (Admin)
    - Retirer un membre

4.6 Paris

- [ ] **4.6.1** `GET /api/bets`
    - Query params: ?leagueId=xxx&userId=xxx&status=pending|won|lost
    - Paris de l'utilisateur
- [ ] **4.6.2** `POST /api/bets`
    - Créer un pari
    - Body: { matchId, leagueId, predictionType: 'winner'|'score'|'both_score'|'handicap', predictionValue: JSON, amount: number }
    - Validations:
        - Match non commencé
        - Points suffisants
        - Pas de pari double sur même match/ligue
- [ ] **4.6.3** `GET /api/bets/:id`
    - Détails pari avec match
- [ ] **4.6.4** `DELETE /api/bets/:id`
    - Annuler pari (uniquement avant début match)
- [ ] **4.6.5** `GET /api/bets/stats`
    - Statistiques utilisateur
    - Taux de réussite, total gagné/perdu

⚙ 5. SERVICES MÉTIER

5.1 Service de Calcul de Gains

- [ ] **5.1.1** `calculateOdds(match, predictionType)`
    - Algorithme de calcul de cotes
    - Basé sur: historique équipes, classement, domicile/extérieur
- [ ] **5.1.2** `calculatePotentialWin(bet)`
    - Gain potentiel = amount × odds
    - Mise à jour `bet.potentialWin`

5.2 Service de Résolution de Paris

- [ ] **5.2.1** `resolveBetsForMatch(matchId)`
    - Trigger après match terminé
    - Pour chaque pari sur le match:
        - Comparer prédiction vs résultat réel
        - Définir status: 'won' ou 'lost'
        - Calculer `actualWin`
        - Mettre à jour points utilisateur dans `LeagueMember`
    - Logger dans `SyncLog` ou table dédiée
- [ ] **5.2.2** `voidBetsForMatch(matchId)`
    - Si match annulé/reporté

    - Rembourser tous les paris (status='void')
- ☐ **5.2.3** `cronResolveMatches()`
    - Job automatique toutes les 15min
    - Chercher matchs finished non résolus
    - Appeler `resolveBetsForMatch`

5.3 Service de Validation

- ☐ **5.3.1** `validateBetCreation(userId, leagueId, matchId, amount)`
    - Vérifier:
        - Match existe et status="upcoming"
        - User est membre de la ligue
        - User a assez de points
        - Pas de pari duplicate
    - Retourner { valid: boolean, error?: string }
- ☐ **5.3.2** `validateLeagueMembership(userId, leagueId)`
- ☐ **5.3.3** `validateInviteCode(inviteCode)`

🔔 6. NOTIFICATIONS & TEMPS RÉEL

6.1 WebSockets / Server-Sent Events

- ☐ **6.1.1** Installer `socket.io` ou `sse`
- ☐ **6.1.2** Event: `match:started` (match commence)
- ☐ **6.1.3** Event: `match:goal` (but marqué en live)
- ☐ **6.1.4** Event: `match:finished` (match terminé)
- ☐ **6.1.5** Event: `bet:resolved` (pari résolu)

6.2 Notifications Push (Mobile)

- ☐ **6.2.1** Intégrer Firebase Cloud Messaging
- ☐ **6.2.2** Notif: "Ton match commence dans 1h"
- ☐ **6.2.3** Notif: "Tu as gagné 250 points !"
- ☐ **6.2.4** Notif: "Nouveau pari disponible"

📚 7. DOCUMENTATION & TESTS

7.1 Swagger

- ☐ **7.1.1** Documenter tous les nouveaux endpoints
- ☐ **7.1.2** Ajouter schémas pour:
    - Competition, Team, Match, Player, Standing, Bet
    - Requêtes et réponses complètes
- ☐ **7.1.3** Ajouter exemples de réponses

7.2 Tests

- ☐ **7.2.1** Tests unitaires services de sync
- ☐ **7.2.2** Tests unitaires calcul de gains
- ☐ **7.2.3** Tests d'intégration endpoints API
- ☐ **7.2.4** Tests e2e: création pari → résolution → mise à jour points

7.3 Documentation Technique

- ☐ **7.3.1** Guide d'intégration TheSportsDB
- ☐ **7.3.2** Documentation algorithme de cotes
- ☐ **7.3.3** Schéma de l'architecture de sync

🚀 8. OPTIMISATIONS & PRODUCTION

8.1 Performance

- ☐ **8.1.1** Implémenter cache Redis
- ☐ **8.1.2** Indexer colonnes fréquemment queryées
- ☐ **8.1.3** Paginer tous les endpoints de liste
- ☐ **8.1.4** Compresser images (logos/badges) via CDN

8.2 Monitoring

- ☐ **8.2.1** Logger tous les appels TheSportsDB
- ☐ **8.2.2** Alertes si rate limit atteint
- ☐ **8.2.3** Dashboard sync: nb items/jour, erreurs, durée
- ☐ **8.2.4** Métriques utilisateur: paris/jour, gains moyens

8.3 Sécurité

- ☐ **8.3.1** Valider tous les inputs utilisateur
- ☐ **8.3.2** Rate limiting par utilisateur (ex: 100 req/min)
- ☐ **8.3.3** Protéger routes admin avec rôle
- ☐ **8.3.4** Sanitize données TheSportsDB (XSS)

8.4 Évolutivité

- ☐ **8.4.1** Préparer passage à TheSportsDB Premium (si besoin live 2min)
- ☐ **8.4.2** Architecture prête pour ajouter nouveaux sports
- ☐ **8.4.3** Système de feature flags (activer/désactiver compétitions)

9. EXPÉRIENCE UTILISATEUR

9.1 Endpoints Composites (Optimisation frontend)

- ☐ **9.1.1** `GET /api/home/feed`
  - Matchs à venir (7 prochains jours)
  - Matchs live
  - Derniers résultats ligues utilisateur
- ☐ **9.1.2** `GET /api/leagues/:id/dashboard`
  - Classement ligue
  - Paris actifs membres
  - Prochains matchs populaires
- ☐ **9.1.3** `GET /api/matches/:id/bet-context`
  - Détails match
  - Statistiques équipes
  - Cotes calculées
  - Paris déjà placés par l'utilisateur

9.2 Filtres Avancés

- ☐ **9.2.1** `GET /api/matches?myTeams=true`
  - Matchs des équipes favorites utilisateur
- ☐ **9.2.2** `GET /api/competitions/popular`
  - Compétitions avec le plus de paris

MÉTRIQUES DE SUCCÈS

KPIs Techniques

- Taux de synchronisation: >99% (matchs à jour)
- Latence API: <200ms (p95)
- Disponibilité: 99.9%
- Coverage tests: >80%

KPIs Fonctionnels

- 14 compétitions actives
- ~300+ équipes synchronisées
- ~5000+ matchs/saison
- Résolution paris: <5min après fin match

LIVRABLES FINAUX

1. **API REST Complète**
   - 30+ endpoints documentés
   - Multi-sports (7 sports, 14 compétitions)
   - Authentification sécurisée
   - Documentation Swagger à jour
2. **Base de Données**

- Schéma complet avec 10+ modèles
- Migrations versionnées
- Seed de données pour chaque sport

3. **Services de Synchronisation**
   - Sync automatique quotidienne
   - Gestion rate limiting
   - Logging et monitoring
4. **Système de Paris**
   - Création/annulation paris
   - Calcul automatique gains
   - Résolution temps réel
   - Historique et stats
5. **Notifications**
   - WebSockets pour live
   - Push mobile pour événements clés
6. **Tests & Documentation**
   - Tests unitaires + intégration
   - Documentation technique
   - Guide API (Swagger)

SOURCES & RÉFÉRENCES

- TheSportsDB API Documentation
- TheSportsDB Free API
- TheSportsDB Sports Leagues
- API Data Examples

Document vivant - À mettre à jour au fur et à mesure de l'avancement du projet