DOCUMENTO DE ARQUITECTURA BACKEND

1. Documentación de Módulos y Responsabilidades

AppModule

El AppModule constituye el núcleo central del backend y opera como el punto de ensamblaje de todos los módulos funcionales del sistema. Su propósito principal es inicializar la configuración global utilizando ConfigModule, cargar las variables de entorno requeridas por el sistema y establecer la conexión con la base de datos MongoDB mediante MongooseModule, garantizando así que el resto de componentes puedan operar de manera estable. Asimismo, registra middlewares globales como RequestIdMiddleware y LoggingMiddleware, esenciales para proveer trazabilidad y facilitar la depuración desde el frontend. Aunque no define entidades propias, es responsable de integrar módulos como usuarios, equipos, ligas, partidos, jugadores, noticias, favoritos y estadísticas. Para el frontend, el AppModule garantiza la disponibilidad, uniformidad y consistencia de los endpoints, resolviendo así el problema del dominio relacionado con la necesidad de centralizar la infraestructura técnica de forma ordenada y escalable.

CommonModule

El CommonModule concentra todos los elementos transversales de la arquitectura del backend que permiten garantizar seguridad, trazabilidad y uniformidad operativa. Su propósito principal es ofrecer autenticación y autorización mediante AuthGuard, asignar identificadores únicos a las solicitudes a través de RequestIdMiddleware y generar registros estructurados de cada petición mediante LoggingMiddleware. Además, proporciona decoradores personalizados como @Public, @CurrentUser y @RequestId, los cuales permiten un manejo más expresivo del contexto de cada request. Aunque no contiene entidades de negocio, habilita el correcto funcionamiento de todos los módulos que sí requieren protección mediante JWT. Desde el frontend, este módulo posibilita el envío de peticiones autenticadas y mejora la experiencia de depuración mediante requestId. El problema que resuelve es la necesidad de contar con una capa de seguridad estandarizada que pueda aplicarse de forma uniforme sin duplicar lógica en cada módulo del sistema.

UsersModule

El UsersModule administra la identidad digital de los usuarios mediante la entidad User, que contiene información clave como username, email, password en formato hash, rol, avatar y equipo favorito. Su propósito es proporcionar servicios esenciales como registro, autenticación, consulta del perfil mediante JWT, actualización de información personal y eliminación de usuarios. Desde el frontend, este módulo soporta las pantallas de registro, login y perfil, convirtiéndose en el núcleo de la experiencia personalizada. Además, provee el token JWT utilizado por los demás módulos para validar el acceso a contenido privado. El problema del dominio que resuelve es la necesidad de un sistema de autenticación robusto que brinde seguridad, administración de roles, control de acceso y persistencia de datos

personales, permitiendo diferenciar usuarios y adaptar la experiencia al contexto individual de cada uno.

TeamsModule

El TeamsModule gestiona la entidad Team, encargada de almacenar información relevante como nombre del equipo, liga, temporada, posición, puntos, goles y racha de rendimiento. Su propósito es proporcionar una estructura completa que permita consultar equipos individuales, listarlos por liga o temporada y generar standings actualizados, utilizando datos ordenados y normalizados. En el frontend, este módulo alimenta las pantallas de tabla de posiciones, sección de equipos, detalle del club y diversos componentes relacionados con estadísticas y partidos. El problema del dominio que resuelve es la necesidad de representar de manera organizada y eficiente el desempeño competitivo de los equipos dentro de competiciones específicas, facilitando la interpretación del rendimiento deportivo y la comparación entre clubes.

LeaguesModule

El LeaguesModule administra la entidad League, que representa las competiciones deportivas con atributos como identificador lógico, nombre, país, temporada activa, jornada actual y fechas relevantes del torneo. Su propósito es ofrecer un marco organizativo que permite agrupar partidos, equipos y estadísticas dentro de una misma estructura competitiva. El frontend consume este módulo para desplegar selectores de ligas, contextualizar standings, filtrar datos deportivos y mostrar información sobre la temporada vigente. El problema de dominio que resuelve consiste en establecer una categorización clara de las competiciones para que el resto de los módulos pueda operar con filtros consistentes y el usuario pueda interpretar la información dentro del contexto adecuado.

MatchesModule

El MatchesModule maneja la entidad Match, que contiene toda la información relativa a partidos deportivos: equipos participantes, fecha, marcador, estado del partido, jornada, sede y árbitro. Su propósito es permitir la gestión completa de encuentros deportivos, incluyendo la creación, consulta, filtrado por equipo, clasificación en próximos, en vivo o recientes, y actualización del resultado. Desde el frontend, este módulo es esencial para mostrar calendarios, secciones de partidos en vivo, resultados recientes y vistas detalladas de cada encuentro. El problema del dominio que resuelve es estructurar la información de los partidos de una manera flexible, consultable y escalable, ofreciendo datos necesarios para entender el calendario competitivo y el desempeño de los equipos a lo largo de la temporada.

PlayersModule

El PlayersModule administra la entidad Player, encargada de representar toda la información individual relevante de cada jugador, incluyendo su nombre, nacionalidad, posición, características físicas, número, valor de mercado y el equipo al que pertenece. Su propósito es permitir la creación, consulta, búsqueda, actualización y eliminación de jugadores, además de obtener plantillas organizadas por equipo. El frontend lo utiliza para

mostrar la composición de un equipo, acceder a la ficha detallada de un jugador y alimentar componentes relacionados con estadísticas y noticias. El problema del dominio que resuelve es la necesidad de representar de forma estructurada la información individual de los jugadores, permitiendo realizar análisis, comparaciones y búsquedas eficientes dentro de un ecosistema deportivo complejo.

NewsModule

El NewsModule gestiona la entidad News, utilizada para almacenar contenido editorial complementario al desempeño deportivo, como noticias, crónicas, reportes de partidos y artículos informativos. La entidad contiene título, slug, contenido, categoría, etiquetas, equipos relacionados y fecha de publicación. Su propósito es integrar contenido narrativo con la base de datos deportiva, permitiendo contextualizar la información de partidos, jugadores y equipos. En el frontend, este módulo alimenta la vista general de noticias, la sección de artículos destacados, noticias vinculadas a equipos y páginas de detalle. El problema del dominio que resuelve es enriquecer la experiencia del usuario mediante información cualitativa que complementa los datos numéricos de las estadísticas y el rendimiento deportivo.

FavoritesModule

El FavoritesModule administra la entidad Favorite, que asocia a cada usuario autenticado con sus equipos y jugadores favoritos. Su propósito es brindar una capa de personalización que permita almacenar preferencias y generar vistas adaptadas al usuario. Desde el frontend, este módulo habilita la construcción de la pantalla “Mis favoritos” y permite activar o desactivar favoritos mediante botones interactivos. El problema del dominio que resuelve es la necesidad de una experiencia personalizada que permita a cada usuario acceder rápidamente al contenido que considera más relevante, sin tener que realizar búsquedas repetitivas.

StatisticsModule

El StatisticsModule maneja la entidad Statistic, que contiene métricas detalladas del rendimiento de jugadores y equipos como goles, asistencias, minutos jugados, tarjetas, intercepciones y rating general. Su propósito es proporcionar rutas que permitan obtener estadísticas por jugador, estadísticas por equipo, rankings de goleadores y rankings de asistidores. Desde el frontend, alimenta secciones analíticas y comparativas que requieren datos agregados y normalizados. El problema del dominio que resuelve es la necesidad de contar con información objetiva que permita medir desempeño, comparar jugadores o equipos y ofrecer una capa analítica sobre los datos deportivos.

2. Documentación Completa de Endpoints

2.1 UsersModule

El UsersModule se encarga de la autenticación y del ciclo de vida del usuario. Desde el frontend se utiliza en las pantallas de registro, inicio de sesión, gestión de perfil y eliminación de cuenta.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
POST	/users/register	Body	RegisterUserDto	Registro desde formulario de signup
POST	/users/login	Body	LoginUserDto	Pantalla de login; genera JWT para el resto del sistema
GET	/users/me	Header Authorization: Bearer <token>	-	Carga perfil del usuario logueado
PATCH	/users/:id	Path id + Body	UpdateUserDto	Edición de perfil
DELETE	/users/:id	Path id	-	Flujo de eliminación de cuenta

DTOs principales:

RegisterUserDto: { username: string; email: string; password: string; firstName?: string; lastName?: string }

LoginUserDto: { email: string; password: string }

UpdateUserDto: campos opcionales para actualizar firstName, lastName, avatar, favoriteTeamId.

Ejemplo de request/response – POST /users/login

Request (frontend envía credenciales desde el formulario de login):

```
{
 "email": "samara@example.com",
 "password": "MiClaveSegura123"
}
```

Response:

```
{
 "user": {
  "id": "65af1234...",
```

```
  "username": "samara",
  "email": "samara@example.com",
  "role": "user"
 },
 "token": "eyJhbGciOi..."
}
```


Uso desde el frontend: el token se almacena (por ejemplo en localStorage) y se envía en el header Authorization: Bearer <token> en todas las peticiones protegidas para acceder al resto de módulos.

2.2 TeamsModule

El TeamsModule administra la información de equipos y standings. El frontend lo usa en la tabla de posiciones, listados de equipos y pantalla de detalle de club.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
POST	/teams	Body	CreateTeamDto	Crear equipo desde panel de administración
GET	/teams	Query leagueId?, season?	-	Listar equipos filtrando por liga/temporada
GET	/teams/search	Query q	-	Buscador de equipos por nombre
GET	/teams/standings/:leagueId	Path leagueId + Query season?	-	Tabla de posiciones por liga
GET	/teams/:id	Path id	-	Detalle de un equipo específico
PATCH	/teams/:id	Path id + Body	CreateTeamDto parcial	Actualizar datos del equipo
DELETE	/teams/:id	Path id	-	Eliminar

				equipo

DTO principal:

CreateTeamDto: { name: string; logo: string; leagueId: string; season: string; position: number; points: number; played: number; wins: number; draws: number; losses: number; goalsFor: number; goalsAgainst: number; goalDifference: number; form: string }

Ejemplo de request/response – GET /teams/standings/:leagueId

Request (frontend consulta standings de una liga):

GET /teams/standings/laliga?season=2024-2025

Response:


```
[
 {
  "id": "t01",
  "name": "Real Madrid",
  "leagueId": "laliga",
  "season": "2024-2025",
  "position": 1,
  "points": 45,
  "played": 19,
  "wins": 14,
  "draws": 3,
  "losses": 2,
  "goalsFor": 42,
  "goalsAgainst": 20,
  "goalDifference": 22,
  "form": "WWDLW"
 }
]
```


Uso desde el frontend: la respuesta se mapea a filas de la tabla de posiciones, mostrando columnas como posición, nombre del equipo, partidos jugados, victorias, empates, derrotas, diferencia de gol y puntos.

2.3 LeaguesModule

El LeaguesModule expone endpoints para la gestión de ligas. El frontend utiliza estos datos para llenar selectores de liga y contextualizar standings, partidos y estadísticas.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend

POST	/leagues	Body	CreateLeagueDto	Crear una nueva liga desde panel admin
GET	/leagues	-	-	Listar ligas disponibles en filtros del frontend
GET	/leagues/:leagueId	Path leagueId	-	Mostrar detalle de una liga concreta
PATCH	/leagues/:leagueId	Path leagueId + Body	-	Actualizar datos de una liga
DELETE	/leagues/:leagueId	Path leagueId	-	Eliminar liga

DTO principal:

CreateLeagueDto: { leagueId: string; name: string; country: string; logo?: string; season: string; startDate?: Date; endDate?: Date; currentMatchday: number; totalMatchdays?: number }

Uso desde el frontend: el endpoint GET /leagues se consume para llenar combos de selección de ligas en las vistas de standings, partidos y estadísticas.

2.4 MatchesModule

El MatchesModule gestiona los partidos a través de endpoints que permiten crear, listar, filtrar por estado o equipo, obtener detalles y actualizar resultados. El frontend lo usa para mostrar calendarios, secciones de partidos en vivo, próximos y recientes.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
POST	/matches	Body	CreateMatchDto	Crear un partido desde herramientas de administración
GET	/matches	Query opcionales	-	Listar todos los partidos

GET	/matches/upcoming	Query limit?	-	Mostrar próximos partidos en la página principal
GET	/matches/live	-	-	Sección de partidos en vivo
GET	/matches/recent	Query limit?	-	Listado de partidos recientes
GET	/matches/team/:teamId	Path teamId	-	Calendario de un equipo específico
GET	/matches/:id	Path id	-	Detalle de partido con datos completos
PATCH	/matches/:id	Path id + Body	-	Actualizar datos o resultado de partido
DELETE	/matches/:id	Path id	-	Eliminar partido

DTO principal:

CreateMatchDto: { homeTeamId: string; awayTeamId: string; leagueId: string; matchDate: Date; status: string; homeScore?: number; awayScore?: number; matchday?: number; venue?: string; referee?: string }

Ejemplo de request/response – GET /matches/upcoming

Request:

GET /matches/upcoming?limit=5

Response:


```
[
 {
  "id": "m001",
```

```
  "homeTeamId": "t01",
  "awayTeamId": "t02",
  "leagueId": "laliga",
  "matchDate": "2025-03-10T18:00:00.000Z",
  "status": "scheduled",
  "matchday": 20,
  "venue": "Santiago Bernabéu"
 }
]
```


Uso desde el frontend: se muestran tarjetas o filas con fecha, hora y nombres de los equipos, obtenidos resolviendo los IDs de equipos mediante el módulo de Teams.

2.5 PlayersModule

El PlayersModule administra la información de los jugadores. Permite crear, listar, buscar, consultar por equipo, obtener detalle de un jugador y actualizar o eliminar registros. En el frontend, estos endpoints se consumen para construir la plantilla de cada equipo, mostrar perfiles individuales de jugadores y alimentar secciones de estadísticas y noticias relacionadas con deportistas concretos.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
POST	/players	Body	CreatePlayerDto	Crear jugador desde panel admin
GET	/players	Query opcionales	-	Listar jugadores en vistas generales
GET	/players/search	Query q	-	Buscador de jugadores por nombre
GET	/players/team/:teamId	Path teamId	-	Mostrar plantilla completa de un equipo
GET	/players/:id	Path id	-	Ficha detallada de un jugador

PATCH	/players/:id	Path id + Body	-	Actualizar datos del jugador
DELETE	/players/:id	Path id	-	Eliminar jugador

DTO principal:

CreatePlayerDto: { name: string; firstName?: string; lastName?: string; photo?: string; nationality: string; dateOfBirth?: Date; position: string; jerseyNumber?: number; teamId: string; height?: number; weight?: number; preferredFoot?: string; marketValue?: number }

Ejemplo de request/response – GET /players/team/:teamId

Request:

GET /players/team/65b0abcd1234

Response:


```
[
 {
  "id": "p001",
  "name": "Jugador 1",
  "position": "MID",
  "jerseyNumber": 8,
  "nationality": "Colombia",
  "teamId": "65b0abcd1234"
 },
 {
  "id": "p002",
  "name": "Jugador 2",
  "position": "FWD",
  "jerseyNumber": 9,
  "nationality": "Argentina",
  "teamId": "65b0abcd1234"
 }
]
```


Uso desde el frontend: la respuesta se utiliza para renderizar la lista de jugadores en la página de plantilla del equipo, mostrando nombre, dorsal, posición y otros datos en tarjetas o tablas.

2.6 NewsModule

El NewsModule gestiona las noticias y artículos relacionados con equipos y partidos. Permite crear noticias, listarlas paginadas, consultar noticias destacadas, obtener noticias

por equipo y acceder a una noticia concreta mediante su slug. El frontend lo utiliza en la sección de noticias generales, en carruseles de noticias destacadas y en vistas detalladas asociadas al contexto de un equipo o partido.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
POST	/news	Body	CreateNewsDto	Crear noticia desde interfaz de gestión de contenido
GET	/news	Query limit, skip	-	Listar noticias paginadas en sección principal
GET	/news/featured	-	-	Obtener noticias destacadas para carrusel o portada
GET	/news/team/:teamId	Path teamId	-	Listar noticias asociadas a un equipo
GET	/news/:slug	Path slug	-	Cargar detalle de una noticia
PATCH	/news/:id	Path id + Body	-	Actualizar noticia
DELETE	/news/:id	Path id	-	Eliminar noticia

DTO principal:

CreateNewsDto: { title: string; slug: string; summary: string; content: string; coverImage?: string; category: string; tags?: string[]; relatedTeamIds?: string[]; relatedMatchId?: string; author?: string; publishedAt?: Date; featured?: boolean }

Ejemplo de request/response – GET /news/featured

Request:

GET /news/featured

Response:


```
[
 {
  "id": "n001",
  "title": "Victoria histórica en el clásico",
  "slug": "victoria-historica-clasico",
  "summary": "Un partido decidido en el último minuto.",
  "coverImage": "/images/news/clasico.png",
  "category": "match-report",
  "publishedAt": "2025-03-02T21:00:00.000Z",
  "featured": true
 }
]
```


Uso desde el frontend: la lista devuelta se utiliza para mostrar tarjetas destacadas en la página de inicio o en la sección de noticias, enlazando cada tarjeta a la ruta de detalle basada en el slug.

2.7 FavoritesModule

El FavoritesModule administra los favoritos de cada usuario autenticado. Permite consultar el conjunto actual de equipos y jugadores favoritos, así como agregar o eliminar favoritos. Desde el frontend se usa para construir la vista “Mis favoritos” y para implementar botones tipo estrella que permiten marcar o desmarcar equipos y jugadores.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
GET	/favorites/me	Header Authorization: Bearer <token>	-	Obtener favoritos del usuario logueado
POST	/favorites/teams/:teamId	Path teamId	-	Agregar un equipo a favoritos
DELETE	/favorites/teams/:teamId	Path teamId	-	Quitar un equipo de favoritos
POST	/favorites/players/:playerId	Path playerId	-	Agregar un jugador a favoritos

DELETE	/favorites/players/:playerId	Path playerId	-	Quitar un jugador de favoritos

Entidad principal:

Favorite: { userId: string; favoriteTeams: string[]; favoritePlayers: string[] } (el userId se obtiene del JWT, no desde el frontend).

Ejemplo de request/response – GET /favorites/me

Request (el frontend envía el JWT):

GET /favorites/me
Header: Authorization: Bearer eyJhbGciOi...

Response:

```
{
 "userId": "65af1234...",
 "favoriteTeams": ["t01", "t05"],
 "favoritePlayers": ["p10", "p22"]
}
```

Uso desde el frontend: esta información se usa para marcar con estrella los equipos y jugadores ya favoritos, y para renderizar la página “Mis favoritos” con tarjetas de equipos y jugadores preferidos.

2.8 StatisticsModule

El StatisticsModule ofrece endpoints para crear y consultar estadísticas de rendimiento, tanto a nivel de jugador como de equipo, además de rankings como goleadores y asistidores. El frontend lo utiliza en secciones analíticas, tablas de líderes y vistas de detalle con métricas agregadas.

Método	Ruta	Parámetros	DTO	Descripción / Uso en frontend
POST	/statistics	Body	CreateStatisticDto	Crear o cargar estadísticas desde procesos administrativos
GET	/statistics/player/:playerId	Path playerId	-	Ver estadísticas completas de un jugador

GET	/statistics/team/:teamId	Path teamId + Query season?	-	Estadísticas agregadas de un equipo
GET	/statistics/top-scorers	Query leagueId?, season?, limit?	-	Ranking de goleadores por liga/temporada
GET	/statistics/top-assisters	Query leagueId?, season?, limit?	-	Ranking de asistidores por liga/temporada
PATCH	/statistics/:id	Path id + Body	-	Actualizar registro de estadísticas
DELETE	/statistics/:id	Path id	-	Eliminar estadísticas

DTO principal:

CreateStatisticDto: { playerId: string; teamId: string; season: string; leagueId: string; matchesPlayed?: number; minutesPlayed?: number; goals?: number; assists?: number; yellowCards?: number; redCards?: number; tackles?: number; interceptions?: number; passAccuracy?: number; rating?: number }

Ejemplo de request/response – GET /statistics/top-scorers

Request:

GET /statistics/top-scorers?leagueId=laliga&season=2024-2025&limit=5

Response:


```
[
 {
  "playerId": "p001",
  "teamId": "t01",
  "season": "2024-2025",
  "leagueId": "laliga",
  "matchesPlayed": 20,
  "goals": 18,
  "assists": 4,
  "rating": 7.9
```

```
 }
]
```

Uso desde el frontend: la respuesta se muestra en una tabla de goleadores en la vista de estadísticas, uniendo los IDs de jugadores y equipos con sus nombres para mostrar información legible al usuario.

3. Modelo de Datos / Esquemas del Backend

3.1 Entidad: User

Archivo: users/schemas/user.schema.ts

Descripción general: La entidad User representa a los usuarios registrados en el sistema. Este modelo soporta autenticación, personalización y control de acceso mediante roles. Incluye información básica del usuario, credenciales, avatar y su equipo principal favorito.

Campos y tipos:

- username: string — requerido, único.

- email: string — requerido, único.

- password: string — requerido.

- firstName: string — opcional.

- lastName: string — opcional.

- avatar: string — opcional.

- role: user | admin | moderator — valor por defecto: user.

- favoriteTeamId: ObjectId ref(Team).

Relaciones:

- favoriteTeamId → Team.

Justificación del diseño:

El modelo User se mantiene ligero para velocidad de lectura en autenticación y personalización.

3.2 Entidad: Favorite

Archivo: favorites/schemas/favorite.schema.ts

Descripción general: Documento que contiene los favoritos de un usuario.

Campos:

- userId: ObjectId ref(User) — requerido, único.

- favoriteTeams: ObjectId[] ref(Team).

- favoritePlayers: ObjectId[] ref(Player).

Relaciones:

- userId → User.

- favoriteTeams → Team.

- favoritePlayers → Player.

Justificación:

Separar los favoritos mejora rendimiento y evita inflar la entidad User.

3.3 Entidad: League

Archivo: leagues/schemas/league.schema.ts

Descripción general: Representa competiciones como LaLiga o Premier League.

Campos:

- leagueId: string — requerido, único.

- name: string — requerido.

- country: string — requerido.

- logo: string.

- season: string — requerido.

- startDate: Date.

- endDate: Date.

- currentMatchday: number — por defecto 1.

- totalMatchdays: number.

Relaciones:

- Usado como clave lógica por Match y Statistic.

Justificación:

El uso de leagueId facilita filtros en el frontend y uso de datos externos.

3.4 Entidad: Team

Archivo: teams/schemas/team.schema.ts

Descripción general: Representa un equipo deportivo con estadísticas usadas en standings.

Campos:

- name: string — requerido.

- logo: string — requerido.

- leagueId: string — requerido.

- season: string — requerido.

- position, points, played, wins, draws, losses.

- goalsFor, goalsAgainst, goalDifference.

- form: string.

Índices:

- (leagueId, season).

- (points desc, goalDifference desc).

Relaciones:

- leagueId → League.

- Referenciado por Player, Match y Statistic.

Justificación:

Optimizado para standings y rendimiento deportivo.

3.5 Entidad: Match

Archivo: matches/schemas/match.schema.ts

Descripción general: Representa partidos deportivos con sus datos principales.

Campos:

- homeTeamId: ObjectId ref(Team) — requerido.

- awayTeamId: ObjectId ref(Team) — requerido.

- leagueId: string — requerido.

- matchDate: Date — requerido.

- status: scheduled | live | finished.

- homeScore, awayScore.

- matchday: number.

- venue: string.

- referee: string.

Índice:

- (matchDate, status).

Relaciones:

- homeTeamId → Team.

- awayTeamId → Team.

- leagueId coincide con League.

Justificación:

Soporta vistas como calendario, partidos en vivo y resultados recientes.

3.6 Entidad: Player

Archivo: players/schemas/player.schema.ts

Descripción general: Representa a un jugador profesional.

Campos:

- name: string — requerido.

- firstName, lastName.

- nationality: string — requerido.

- dateOfBirth: Date.

- position: GK | DEF | MID | FWD — requerido.

- jerseyNumber: number.

- teamId: ObjectId ref(Team) — requerido.

- height, weight.

- preferredFoot: left | right | both.

- marketValue: number.

Relaciones:

- teamId → Team.

- favoritePlayers → Player.

- statistics → Player.

Justificación:

Índice por teamId para optimizar carga de plantillas.

3.7 Entidad: News

Archivo: news/schemas/news.schema.ts

Descripción general: Contenido editorial sobre equipos y partidos.

Campos:

- title: string — requerido.

- slug: string — requerido, único.

- summary: string — requerido.

- content: string — requerido.

- coverImage: string.

- category: match-report | transfer | interview | analysis.

- tags: string[].

- relatedTeamIds: ObjectId[] ref(Team).

- relatedMatchId: ObjectId ref(Match).

- author: string.

- publishedAt: Date.

- views: number.

- featured: boolean.

Índices:

- slug.

- (category, publishedAt desc).

Relaciones:

- relatedTeamIds → Team.

- relatedMatchId → Match.

Justificación:

Permite relacionar noticias con eventos y equipos del sistema.

3.8 Entidad: Statistic

Archivo: statistics/schemas/statistic.schema.ts

Descripción general: Guarda estadísticas detalladas de jugadores y equipos.

Campos:

- playerId: ObjectId ref(Player) — requerido.

- teamId: ObjectId ref(Team) — requerido.

- season: string — requerido.

- leagueId: string — requerido.

- matchesPlayed, minutesPlayed.

- goals, assists.

- yellowCards, redCards.

- tackles, interceptions.

- passAccuracy.

- rating.

Índice:

- (playerId, season, leagueId).

Relaciones:

- playerId → Player.

- teamId → Team.

- leagueId → League.

Justificación:

Base para rankings y análisis de rendimiento.

3.9 Resumen General de Relaciones

- User ↔ Favorite (1:1).

- User.favoriteTeamId → Team.

- Favorite.favoriteTeams[] → Team.

- Favorite.favoritePlayers[] → Player.

- Team.leagueId → League.

- Match.homeTeamId, Match.awayTeamId → Team.

- Match.leagueId → League.

- Player.teamId → Team.

- News.relatedTeamIds[] → Team.

- News.relatedMatchId → Match.

- Statistic.playerId → Player.

- Statistic.teamId → Team.

- Statistic.leagueId → League.

4. Díagrama UML