

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN



Sports DB: Diseño e implementación de una
aplicación web y móvil para seguimiento de
resultados de fútbol

TRABAJO FIN DE GRADO

Junio - 2026

AUTOR: Raúl Quesada Vidal

DIRECTOR: Jesús Javier Rodríguez Sala

RESUMEN

SportsDB es una plataforma web y móvil para el seguimiento de resultados de fútbol, desarrollada con Angular 17 en el frontend y Django 5.2 en el backend. La arquitectura sigue el patrón BFF (Backend For Frontend): el cliente Angular nunca llama directamente a las APIs externas, sino que todas las peticiones pasan por el servidor Django, que actúa como intermediario, oculta las credenciales y normaliza los datos antes de devolverlos al cliente.

Los datos deportivos se obtienen en tiempo real desde LiveScore API, proveedor de pago que suministra clasificaciones, resultados, partidos, estadísticas, alineaciones, plantillas, goleadores e historial de enfrentamientos de las principales ligas europeas. La información editorial de equipos y jugadores se enriquece con la Wikipedia REST API, extrayendo extractos, imágenes e infoboxes mediante scraping de wikitext. Los datos propios del sistema —usuarios, favoritos, noticias y minijuego— se persisten en una base de datos MySQL gestionada con el ORM de Django.

La aplicación ofrece clasificaciones con soporte para fases de grupos y eliminatorias, resultados por jornada con selector de temporadas históricas, partidos en directo con marcador en tiempo real, bracket visual de competiciones eliminatorias, fichas completas de equipo con palmarés extraído de Wikipedia, modal de jugadores con datos biográficos, y una sección de favoritos enriquecida que muestra el último resultado y el próximo partido de cada equipo guardado. Complementan la aplicación una sección de noticias administrable desde el panel de Django y un minijuego de trivia diario configurable por el administrador.

El frontend está construido con componentes standalone de Angular 17, RxJS para la gestión de la asincronía, Bootstrap 5 para el diseño responsive y Angular Material para los componentes de interacción. La gestión del estado global se implementa mediante cuatro servicios singleton en la carpeta core/ sin dependencias externas como NgRx. La aplicación incorpora Server-Side Rendering con Angular SSR para mejorar el tiempo de carga inicial. Mediante Capacitor 7.4, el mismo código Angular se compila como APK instalable en dispositivos Android, conectándose al mismo backend Django por red local.

El proyecto incluye 28 requisitos funcionales, 5 requisitos no funcionales, diagramas de casos de uso, modelo entidad-relación, diagramas de componentes Angular, diagramas de secuencia, mockups de las interfaces principales y un manual de usuario completo como Anexo.

AGRADECIMIENTOS

Quisiera comenzar expresando mi más sincero agradecimiento a Jesuja, tutor de este Trabajo de Fin de Grado, por su compromiso, orientación y cercanía durante todo el desarrollo del proyecto. Su implicación y apoyo constante han sido fundamentales para culminar esta etapa con éxito.

Extiendo también mi agradecimiento a la Universidad Miguel Hernández (UMH) por ofrecer un entorno académico en el que he podido crecer personal y profesionalmente, así como al IES La Mar de Jávea, lugar donde descubrí mi pasión por la informática y comprendí que este camino sería el mío. A todos los profesores que han marcado mi trayectoria, gracias por vuestro tiempo, vuestra paciencia y por transmitir el valor del esfuerzo y la curiosidad por aprender. En especial, a Raúl, en mis años de instituto, y a Germán, durante el ciclo formativo de Desarrollo de Aplicaciones Multiplataforma (DAM), por su motivación y enseñanza constante.

A mis amigos —Víctor López, Javier Carrillo, Javier Poquet, Ismael Ferrer y Vicente Oliver—, gracias por acompañarme dentro y fuera de las aulas, por los consejos, las risas y el ánimo en los momentos más intensos del proyecto.

A mis compañeros de piso, Carlos Llorca y Gaspar Cruaños, gracias por compartir este camino, por la comprensión en los días de entrega y por las interminables conversaciones que tanto aportaron.

A mis padres, por ser mi mayor fuente de apoyo, por su confianza incondicional y por enseñarme el valor del trabajo y la perseverancia. Este logro también es vuestro.

Finalmente, quiero agradecer a todos los personajes públicos del mundo del fútbol que, sin saberlo, han sido una fuente constante de inspiración y pasión para mí. Gracias a Isco Alarcón y al Málaga CF, símbolos de esfuerzo, talento y amor por este deporte, por haber despertado en mí una motivación que ha llegado incluso a influir en la creación de este proyecto.

A todos, gracias por formar parte de este viaje.

ÍNDICE GENERAL

1. INTRODUCCIÓN	12
1.1. La informática en el mundo del fútbol	12
1.2. Justificación del proyecto	14
1.2.1. Motivación personal y conexión con el fútbol	15
1.2.2. Interés académico y desarrollo profesional	15
1.2.3. Retos personales y superación técnica	16
1.2.4. Aprendizaje y aplicación de nuevas tecnologías	16
1.2.5. Aporte al sector deportivo y a la comunidad de aficionados	17
1.2.6. Innovación y relevancia del proyecto	17
1.2.7. Importancia social y cultural del fútbol digital	17
1.3. Objetivos	18
1.3.1. Objetivos principales	18
1.3.2. Objetivos secundarios	19
1.4. Límites del proyecto	21
2. ANTECEDENTES Y ESTADO DE LA CUESTIÓN	23
2.1. Situación actual del sector	23
2.2. Herramientas disponibles en el mercado	24
2.2.1. Flashscore	24
2.2.2. SofaScore	24
2.2.3. LiveScore	25
2.2.4. BeSoccer	25
2.2.5. 365Score	25
2.2.6. Resumen comparativo	26
2.3. Valoración	26
3. HIPÓTESIS DE TRABAJO	27
3.1. Tecnologías de frontend	27
3.1.1. Angular 17	27
3.1.1.1. TypeScript	28
3.1.1.2. RxJS	28
3.1.1.3. Angular SSR	28
3.1.2. Bootstrap 5	29
3.1.3. Angular Material	29
3.1.4. Capacitor	29
3.1.4.1. Configuración del proyecto Android	30
3.1.4.2. Flujo de desarrollo en dispositivo físico	30
3.1.4.3. Compilación de la APK y debug con Chrome DevTools	31

3.1.5. ApexCharts	32
3.2. Tecnologías de backend	32
3.2.1. Django 5	32
3.2.2. Django REST Framework	33
3.2.3. Python	33
3.3. Base de datos y almacenamiento	33
3.3.1. MySQL	33
3.3.2. Django ORM	33
3.4. Entorno de desarrollo y servidor	34
3.4.1. XAMPP	34
3.4.2. Node.js y Angular CLI	34
3.4.3. Gestión de entornos y proxy de desarrollo	34
3.4.4. Django como servidor de desarrollo	36
3.4.5. Git y control de versiones	37
3.4.6. Visual Studio Code	37
3.5. APIs externas	37
3.5.1. LiveScore API	37
3.5.1.1. Postman como herramienta de exploración	39
3.5.2. Wikipedia API	40
3.6. Arquitectura BFF y patrón REST	40
3.7. Gestión de seguridad: CORS y CSRF	41
3.8. Inteligencia artificial como herramienta de desarrollo	41
3.9. Resumen del stack tecnológico	42
 4. METODOLOGÍA Y RESULTADOS	 43
4.1. Planificación del proyecto	43
4.1.1. Ciclo de vida del software	43
4.2. Captura de requisitos	45
4.2.1. Tipos de usuarios	45
4.2.2. Requisitos funcionales	46
4.2.3. Requisitos no funcionales	50
4.2.4. Diagramas de casos de uso	51
4.3. Diseño	54
4.3.1. Modelo de datos (Diagrama E/R)	54
4.3.2. Arquitectura del sistema	55
4.3.3. Diagrama de componentes Angular	56
4.3.4. Gestión de estado en el frontend	57
4.3.5. Diagramas de secuencia	59
4.3.5.1. Consulta clasificación	59
4.3.5.2. Añadir equipo a favoritos	59
4.3.6. Diseño de la interfaz gráfica	60
4.4. Implementación	73

4.4.1. Estructura del proyecto y sincronización Angular-Django	73
4.4.1.1. Estructura backend Django	73
4.4.1.2. Estructura frontend Angular	74
4.4.1.3. Comunicación backend-frontend	75
4.4.2. Sistema centralizado de llamadas con fallback	77
4.4.3. Página principal: combinación de tres fuentes	77
4.4.4. Resultados por jornada: fusión de tres estados	79
4.4.5. Detalle de partido con peticiones paralelas	81
4.4.6. Resumen de partido: el problema del home/away	81
4.4.7. Clasificaciones y el problema de los grupos falsos	83
4.4.8. Bracket eliminatorio y detección de fases	84
4.4.9. Normalización de datos inconsistentes de la API	85
4.4.10. Integración de Wikipedia: scraping de palmarés	85
4.4.11. Sección de noticias y minijuego	87
4.4.12. Aplicación Android con Capacitor	89
4.4.13. Tipos de competición y menú dinámico	90
4.4.14. Caché en memoria de datos de competición	90
4.4.15. Dificultades del frontend y soluciones adoptadas	90
4.4.16. Problemas de compatibilidad con Angular SSR	92
4.5. Pruebas e implantación	93
4.5.1. Pruebas funcionales	93
4.5.2. Pruebas con el modo mock	94
4.5.3. Implantación local	94
 5. CONCLUSIONES Y TRABAJO FUTURO	 95
5.1. Conclusiones	95
5.1.1. Conclusiones	95
5.1.2. Valoración personal del proceso	96
5.1.3. Limitaciones y trabajo futuro	97
5.2. Posibles desarrollos futuros	98
5.2.1. Seguridad y producción	98
5.2.2. Mejoras en la gestión de usuarios	98
5.2.3. Ampliación de contenidos deportivos	99
5.2.4. Mejoras de interfaz y experiencia de usuario	99
5.2.5. Ampliaciones de la app Android	99
5.2.6. Módulos nuevos	100
5.2.7. Infraestructura y despliegue	100
 6. BIBLIOGRAFÍA	 101
6.1. Frameworks y lenguajes de programación	101
6.2. Librerías y herramientas de frontend	102
6.3. Plataformas móviles	103

6.4. APIs externas	103
6.5. Bases de datos y entornos de desarrollo	103
6.6. Arquitectura y patrones de diseño	104
6.7. Ingeniería del software y metodología	105
6.8. Plataformas de referencia y antecedentes	105
6.9. Tecnologías web complementarias	106
ANEXO I — Casos de Uso	107
ANEXO II — Manual de usuario	122
AII.1. Interfaz general	123
AII.2. Página de inicio	124
AII.3. Seleccionar una liga	124
AII.4. Clasificación	125
AII.5. Resultados	126
AII.6. Partidos próximos	128
AII.7. En directo	128
AII.8. Detalle de partido	129
AII.8.1. Resumen (eventos)	129
AII.8.2. Estadísticas	130
AII.8.3. Alineaciones	130
AII.8.4. H2H (Head to Head)	131
AII.9. Bracket eliminatorio	132
AII.10. Ficha de equipo	132
AII.10.1. Plantilla	133
AII.10.2. Resultados	134
AII.10.3. Estadísticas	135
AII.10.4. Información	136
AII.11. Registro de usuario	136
AII.12. Inicio de sesión	137
AII.13. Cerrar sesión	138
AII.14. Equipos favoritos	138
AII.14.1. Añadir equipo a favoritos	138
AII.14.2. Eliminar un equipo de favoritos	139
AII.14.3. Ver la lista de favoritos	139
AII.15. Sección de noticias	140
AII.16. Minijuego del día	141
AII.17. Panel de administración	143
AII.17.1. Gestión de noticias	143
AII.17.2. Gestión de minijuego	144
AII.17.3. Gestión de usuarios	145
AII.18. Aplicación Android	145

ÍNDICE DE TABLAS

Tabla 2.1	Resumen comparativo competencia	26
Tabla 3.1.	Tabla de entornos Angular (web y Android)	35
Tabla 3.2.	Resumen del stack tecnológico de SportsDB	42
Tabla 4.1.	Tipo de usuario anónimo	45
Tabla 4.2.	Tipo de usuario identificado	45
Tabla 4.3.	Tipo de usuario administrador	45
Tabla 4.4.	RF-1 Visualización de la página de inicio	46
Tabla 4.5.	RF-2 Selector de fecha en la página de inicio	46
Tabla 4.6.	RF-3 Visualización de clasificaciones	46
Tabla 4.7.	RF-4 Clasificaciones históricas por temporada	46
Tabla 4.8.	RF-5 Visualización de resultados por jornada	46
Tabla 4.9.	RF-6 Selector de jornada	47
Tabla 4.10.	RF-7 Visualización de partidos próximos	47
Tabla 4.11.	RF-8 Partidos en directo	47
Tabla 4.12.	RF-9 Goleadores de una competición	47
Tabla 4.13.	RF-10 Eventos de un partido	47
Tabla 4.14.	RF-11 Estadísticas de un partido	47
Tabla 4.15.	RF-12 Alineaciones de un partido	48
Tabla 4.16.	RF-13 Head to Head entre dos equipos	48
Tabla 4.17.	RF-14 Bracket eliminatorio	48
Tabla 4.18.	RF-15 Detalle de equipo	48
Tabla 4.19.	RF-16 Información de jugador	48
Tabla 4.20.	RF-17 Registro de usuario	48
Tabla 4.21.	RF-18 Inicio y cierre de sesión	49
Tabla 4.22.	RF-19 Gestión de equipos favoritos	49
Tabla 4.23.	RF-20 Vista de favoritos enriquecida	49
Tabla 4.24.	RF-21 Sección de noticias	49
Tabla 4.25.	RF-22 Minijuego de trivia	49
Tabla 4.26.	RF-23 Sidebar de navegación por ligas	49
Tabla 4.27.	RF-24 Navegación con breadcrumb	50
Tabla 4.28.	RF-25 Gestión de noticias desde panel de administración	50
Tabla 4.29.	RF-26 Gestión del minijuego desde panel de administración	50
Tabla 4.30.	RF-27 Gestión de usuarios desde panel de administración	50
Tabla 4.31.	RF-28 Aplicación Android	50
Tabla 4.32.	RNF-1 Rendimiento	51
Tabla 4.33.	RNF-2 Usabilidad	51
Tabla 4.34.	RNF-3 Seguridad	51
Tabla 4.35.	RNF-4 Disponibilidad del modo offline	51

Tabla 4.36. RNF-5 Responsabilidad	51
Tabla 4.37. Servicios singleton de estado en SportsDB	58
Tabla 4.38. Compatibilidad SSR — problemas y soluciones	92
Tabla AI.1. C.U.-1 Ver partidos del día	108
Tabla AI.2. C.U.-2 Navegar por fechas en la página de inicio	108
Tabla AI.3. C.U.-3 Ver clasificación de una liga	109
Tabla AI.4. C.U.-4 Consultar clasificación de temporada anterior	109
Tabla AI.5. C.U.-5 Ver resultados de una jornada	110
Tabla AI.6. C.U.-6 Navegar entre jornadas	110
Tabla AI.7. C.U.-7 Ver partidos próximos	111
Tabla AI.8. C.U.-8 Ver partidos en directo	111
Tabla AI.9. C.U.-9 Ver goleadores de una liga	112
Tabla AI.10. C.U.-10 Ver eventos de un partido	112
Tabla AI.11. C.U.-11 Ver estadísticas de un partido	113
Tabla AI.12. C.U.-12 Ver alineaciones de un partido	113
Tabla AI.13. C.U.-13 Ver historial H2H entre dos equipos	114
Tabla AI.14. C.U.-14 Ver bracket eliminatorio	114
Tabla AI.15. C.U.-15 Ver ficha de equipo	115
Tabla AI.16. C.U.-16 Ver ficha de jugador	115
Tabla AI.17. C.U.-17 Registrarse	116
Tabla AI.18. C.U.-18 Iniciar Sesión	116
Tabla AI.19. C.U.-19 Cerrar sesión	117
Tabla AI.20. C.U.-20 Añadir equipo a favoritos	117
Tabla AI.21. C.U.-21 Eliminar equipo de favoritos	118
Tabla AI.22. C.U.-22 Ver favoritos enriquecidos	118
Tabla AI.23. C.U.-23 Ver sección de noticias	119
Tabla AI.24. C.U.-24 Seleccionar liga en el sidebar	119
Tabla AI.25. C.U.-25 Jugar el minijuego del día	120
Tabla AI.26. C.U.-26 Gestionar noticias desde el panel de administración	120
Tabla AI.27. C.U.-27 Gestionar el minijuego desde el panel de administración	121
Tabla AI.28. C.U.-28 Gestionar usuarios desde el panel de administración	121

ÍNDICE DE FIGURAS

Figura 3.1. Logotipo de Angular	28
Figura 3.2. Logotipo de Capacitor	29
Figura 3.3. Logotipo de Django	32
Figura 3.4. Logotipo de Git	37
Figura 3.5. Logotipo de LiveScore API	38
Figura 3.6. Captura de la colección de LiveScore API en Postman	39
Figura 3.7. Logotipo de Wikipedia	40
Figura 3.8. Logotipo de Claude	41
Figura 4.1. Diagrama de Gantt del proyecto	44
Figura 4.2. Jerarquía de tipos de usuario	45
Figura 4.3. Diagrama de casos de uso — Usuario anónimo	52
Figura 4.4. Diagrama de casos de uso — Usuario identificado	53
Figura 4.5. Diagrama de casos de uso — Administrador	53
Figura 4.6. Diagrama Entidad-Relación de SportsDB	54
Figura 4.7. Diagrama de arquitectura BFF del sistema	55
Figura 4.8. Diagrama de componentes del frontend Angular	56
Figura 4.9. Diagrama de secuencia — Consulta de clasificación	59
Figura 4.10. Diagrama de secuencia — Añadir equipo a favoritos	60
Figura 4.11. Mockup — Página de inicio	61
Figura 4.12. Mockup — Clasificación	62
Figura 4.13. Mockup — Resultados por jornada	63
Figura 4.14. Mockup — Detalle de partido (eventos)	64
Figura 4.15. Mockup — Ficha de equipo (información Wikipedia)	65
Figura 4.16. Mockup — Plantilla de equipo	66
Figura 4.17. Mockup — Bracket eliminatorio	67
Figura 4.18. Mockup — Favoritos enriquecidos	68
Figura 4.19. Mockup — Login y Registro	69
Figura 4.20. Mockup — Noticias	70
Figura 4.21. Mockup — Minijuego del día	71
Figura 4.22. Mockup — Vistas móviles Android	72
Figura 4.23. Árbol de directorios y archivos backend	73
Figura 4.24. Árbol de directorios y archivos frontend	75
Figura 4.25. Esquema del flujo de <code>api_call_with_fallback</code>	76
Figura 4.26. Captura de la página principal de SportsDB	79
Figura 4.27. Captura de resultados por jornada	80
Figura 4.28. Captura del detalle de equipo — Wikipedia	81
Figura 4.29. Captura del detalle de partido — tab eventos	83
Figura 4.30. Captura de la clasificación con grupos genuinos	84

Figura 4.31. Captura del bracket de la Eurocopa	85
Figura 4.32. Captura del modal de jugador	87
Figura 4.33. Captura del panel de administración Django	88
Figura 4.34. Captura del minijuego de trivia	88
Figura 4.35. Captura de la aplicación Android en dispositivo	89
Figura AII.1. Interfaz general de SportsDB	123
Figura AII.2. Selector de fecha y botón En Directo	124
Figura AII.3. Cabecera de liga con pestañas de navegación	125
Figura AII.4. Tabla de clasificación con posiciones coloreadas	125
Figura AII.5. Selector de temporada en clasificación histórica	126
Figura AII.6. Pestaña de goleadores	126
Figura AII.7. Sección de resultados por fecha y estado	127
Figura AII.8. Selector de jornada desplegable	127
Figura AII.9. Sección de partidos próximos	128
Figura AII.10. Sección de partidos en directo	128
Figura AII.11. Ficha de partido — tab Resumen (eventos)	129
Figura AII.12. Ficha de partido — tab Estadísticas	130
Figura AII.13. Ficha de partido — tab Alineaciones	131
Figura AII.14. Ficha de partido — tab H2H	131
Figura AII.15. Bracket eliminatorio Eurocopa	132
Figura AII.16. Ficha de equipo — tab Plantilla	133
Figura AII.17. Modal de jugador con información Wikipedia	134
Figura AII.18. Ficha de equipo — tab Resultados	135
Figura AII.19. Ficha de equipo — tab Estadísticas	135
Figura AII.20. Ficha de equipo — tab Información Wikipedia	136
Figura AII.21. Formulario de registro	137
Figura AII.22. Formulario de inicio de sesión	137
Figura AII.23. Header con sesión iniciada	138
Figura AII.24. Estrella de favorito — inactivo y activo	138
Figura AII.25. Sección de favoritos	139
Figura AII.26. Tarjeta de favorito expandida	140
Figura AII.27. Sección de noticias	140
Figura AII.28. Vista de detalle de noticia	141
Figura AII.29. Minijuego del día — estado inicial	141
Figura AII.30. Minijuego en progreso — respuestas acertadas	142
Figura AII.31. Panel de administración Django	143
Figura AII.32. Gestión de noticias desde el panel	143
Figura AII.33. Añadir noticias desde el panel	144
Figura AII.34. Configuración del minijuego desde el panel	144
Figura AII.35. Gestión de usuarios desde el panel	145
Figura AII.36. Pantalla de inicio en dispositivo Android	145
Figura AII.37. Navegación inferior en la versión Android	146

ÍNDICE DE CÓDIGOS

Código 3.1. capacitor.config.ts — configuración Live Reload Android	30
Código 3.2. proxy.conf.json — configuración del proxy de desarrollo	35
Código 3.3. Comandos de build y arranque backend frontend	36
Código 4.1. api_call_with_fallback — sistema de fallback a mock	77
Código 4.2. _fetch_history_pages — paginación de history.json	78
Código 4.3. _fetch_all_pages — paginación de fixtures/list.json	80
Código 4.4. _side_by_name — resolución del home/away por nombre	82
Código 4.5. _filter_genuine_groups — filtro de grupos duplicados	83
Código 4.6. _GROUP_STAGE_RE — regex de detección de fase de grupos	84
Código 4.7. Compilación y arranque android	89
Código 4.8. getCsrftoken — lectura segura con isPlatformBrowser	92
Código 4.9. renderChart — guard SSR para ApexCharts	93



Capítulo 1

Introducción

1.1.- LA INFORMÁTICA EN EL MUNDO DEL FÚTBOL

El fútbol se ha consolidado como el deporte más seguido del planeta, capaz de unir a millones de personas bajo una misma pasión. Más allá del aspecto competitivo, se ha convertido en un fenómeno cultural, social y económico que trasciende fronteras, edades y clases sociales. Grandes competiciones como la UEFA Champions League, la Copa del Mundo de la FIFA o las principales ligas nacionales (Premier League, LaLiga, Serie A, Bundesliga o Ligue 1) logran audiencias globales que superan los cientos de millones de espectadores por partido, generando un impacto mediático sin precedentes.

El crecimiento del fútbol moderno no puede entenderse sin la evolución tecnológica que lo acompaña. La expansión de Internet, la mejora en las comunicaciones móviles y la popularización de los dispositivos inteligentes han transformado la manera en que los

aficionados viven este deporte. Lo que antes se limitaba a la televisión y la prensa escrita, hoy se ha trasladado a un entorno digital interactivo y en tiempo real, donde la información fluye constantemente y el usuario participa de forma activa.

Las aplicaciones web y móviles han adquirido un papel protagonista en esta nueva era del fútbol digital. Permiten acceder en cuestión de segundos a resultados, clasificaciones, alineaciones, estadísticas, traspasos, comparativas de rendimiento y noticias actualizadas. Gracias a estas plataformas, el seguidor puede consultar cualquier dato de su equipo o jugador favorito desde cualquier lugar del mundo, convirtiendo el seguimiento deportivo en una experiencia continua y personalizada.

Esta nueva forma de consumir deporte ha generado un cambio profundo en el comportamiento del usuario. El aficionado actual no se conforma con saber el resultado final: quiere conocer los detalles del partido, los datos de posesión, los remates, las tarjetas, el estado físico de los jugadores o incluso la predicción de rendimiento basada en estadísticas avanzadas. Esta tendencia ha dado lugar al auge del *“data-driven football”*, donde la información y el análisis se convierten en elementos clave tanto para los profesionales del deporte como para los espectadores.

Desde la perspectiva del consumidor digital, la accesibilidad y la usabilidad son dos factores determinantes. Las aplicaciones deportivas deben ser intuitivas, visuales y rápidas, capaces de ofrecer una gran cantidad de información sin abrumar al usuario. El diseño responsive, la simplicidad en la navegación y la disponibilidad inmediata de los datos son aspectos esenciales para garantizar una buena experiencia de uso. Un retraso de apenas unos segundos en la actualización de un marcador puede significar una pérdida de interés o credibilidad en un sector donde la inmediatez es fundamental.

Por otra parte, el volumen de usuarios y la afluencia global que generan estas plataformas es impresionante. Según estudios recientes, más del 70 % de los aficionados al fútbol consulta aplicaciones o webs deportivas al menos una vez al día, especialmente durante fines de semana y competiciones internacionales. Este tráfico masivo convierte al sector digital-deportivo en uno de los más activos de Internet, superando incluso a otros ámbitos de entretenimiento como la música o el cine en términos de visitas simultáneas y tiempo de uso.

El fútbol, como industria, también se ha visto beneficiado por esta digitalización. Los clubes, ligas y federaciones utilizan la tecnología para mejorar su comunicación con los aficionados, optimizar la gestión de sus competiciones y aumentar su visibilidad a nivel global. Las redes sociales, los canales de streaming y las plataformas oficiales permiten conectar directamente con los seguidores, fomentando un sentimiento de pertenencia y comunidad. En este contexto, las aplicaciones especializadas se convierten en el punto de

encuentro entre el usuario y el universo futbolístico, sirviendo de puente entre la información oficial, la actualidad y la pasión del aficionado.

Asimismo, el auge de los datos en tiempo real ha abierto un nuevo campo dentro del sector: el análisis y la interpretación de estadísticas. Hoy en día, cada partido genera miles de datos sobre rendimiento, estrategia y condición física de los jugadores. Estos datos, procesados y presentados de forma comprensible, no solo son útiles para los entrenadores y analistas, sino también para los aficionados, que disfrutan entendiendo el juego desde una perspectiva más profunda.

El desarrollo de este tipo de plataformas también está estrechamente relacionado con el crecimiento del periodismo deportivo digital. Los medios de comunicación se apoyan cada vez más en herramientas que proporcionan datos instantáneos y verificables, integrando información procedente de APIs o servicios automatizados en sus publicaciones y retransmisiones. Así, la informática se ha convertido en un pilar esencial para la cobertura y difusión del fútbol, garantizando la rapidez y fiabilidad que exige el público contemporáneo.

En definitiva, el entorno de aplicación en el que se enmarca este proyecto es un espacio donde la tecnología y el deporte convergen para ofrecer una experiencia global, conectada y personalizada. La informática ha dejado de ser un mero apoyo para convertirse en un elemento estructural del ecosistema futbolístico, presente en todas sus dimensiones: desde la gestión de competiciones hasta la relación entre el aficionado y el contenido.

La creación de plataformas como SportsDB responde a esta realidad: la necesidad de centralizar, organizar y presentar la información futbolística de forma accesible, dinámica y atractiva para un público que ya no concibe el deporte sin la mediación de la tecnología. En un mundo donde el fútbol se vive las 24 horas del día, la informática es el instrumento que permite que esa pasión no tenga fronteras ni horarios.

1.2.- JUSTIFICACIÓN DEL PROYECTO

El desarrollo de SportsDB surge como una oportunidad para unir dos mundos que siempre han tenido un gran peso en mi vida: el fútbol y la informática. En un contexto donde las aplicaciones deportivas han adquirido un papel fundamental en la forma en que los usuarios consumen información, este proyecto busca ofrecer una plataforma moderna, visual y estructurada que acerque los datos futbolísticos a los aficionados.

Más allá de un ejercicio técnico, SportsDB representa un reto personal y académico, en el que he podido aplicar los conocimientos adquiridos durante el grado, explorar nuevas

tecnologías y demostrar mi capacidad para desarrollar una aplicación completa desde cero. Su justificación se apoya tanto en una motivación personal y profesional como en una necesidad real dentro del ámbito deportivo: disponer de herramientas web que centralicen información, resultados y estadísticas de forma rápida, accesible y fiable.

1.2.1.- Motivación personal y conexión con el fútbol

Desde muy pequeño, el fútbol ha estado presente en mi vida como algo más que un simple deporte. Siempre he sentido una profunda conexión con todo lo que lo rodea: las emociones, la historia, los jugadores y el ambiente que se vive en torno a cada partido. Jugadores como Isco Alarcón o incluso grandes mitos del pasado, como Maradona, fueron referentes que despertaron en mí una admiración y una curiosidad por comprender el fútbol más allá del resultado.

Mi relación con este deporte no se ha limitado al campo o a la pantalla del televisor, sino también al consumo constante de información. Desde niño me fascinaba estar al tanto de todo lo que ocurría en el mundo futbolístico. Consultaba a diario páginas y periódicos digitales como Marca o AS para conocer los resultados de la jornada, leer crónicas o descubrir los próximos fichajes. Con el tiempo, esa costumbre se convirtió en una verdadera pasión por los datos: me gustaba conocer las plantillas históricas, los traspasos entre clubes, los registros estadísticos de jugadores o las historias de décadas pasadas.

Esa curiosidad y deseo de estar siempre informado me definieron como un aficionado forofo, alguien que disfruta del fútbol en todas sus formas, desde la actualidad hasta su historia. Por eso, el desarrollo de esta aplicación no es solo un ejercicio técnico, sino también la materialización de una pasión personal.

SportsDB encaja conmigo “como anillo al dedo”: une mi interés por la informática con esa necesidad casi natural de seguir y entender el fútbol en profundidad. Poder desarrollar una plataforma que centralice toda esa información que yo mismo he buscado durante años representa no solo un reto académico, sino también una satisfacción personal inmensa.

1.2.2.- Interés académico y desarrollo profesional

El proyecto se concibe como la culminación práctica del aprendizaje adquirido en el Grado en Ingeniería Informática, integrando conceptos de diseño web, bases de datos, arquitectura cliente-servidor y gestión de proyectos.

Gracias a SportsDB, he podido afianzar mi formación como desarrollador full stack, combinando el trabajo en Angular para el frontend con Django en el backend. Además, el

desarrollo de un sistema que se comunica con APIs externas (LiveScore y Wikipedia) ha supuesto una experiencia muy próxima al entorno profesional actual, donde la integración de servicios y la gestión de datos en tiempo real son habilidades altamente demandadas.

En este sentido, el proyecto me ha permitido crecer tanto en lo técnico como en lo personal, consolidando mi perfil como profesional del desarrollo web.

1.2.3.- Retos personales y superación técnica

Cada proyecto de programación supone un nuevo desafío, una oportunidad para mejorar y superarse. En el caso de SportsDB, este reto ha sido especialmente significativo, tanto en el plano técnico como en el personal. Desde el primer momento supe que no sería una aplicación sencilla: requería integrar distintos módulos, manejar una gran cantidad de datos, y coordinar frontend y backend en un entorno complejo.

Sin embargo, más allá de los aspectos puramente técnicos, el verdadero desafío ha estado en la organización y la constancia. Este proyecto ha coincidido con un periodo de prácticas y trabajo, por lo que he tenido que compaginar responsabilidades profesionales con el desarrollo del TFG. Aprendí a distribuir el tiempo de manera eficiente, a planificar tareas por etapas y a mantener la motivación incluso en los momentos de mayor carga.

SportsDB también ha representado una superación personal en términos de aprendizaje. He tenido que enfrentarme a tecnologías nuevas, comprender su funcionamiento desde la base y aplicar buenas prácticas de desarrollo para conseguir un código limpio, escalable y mantenible. Cada error y cada bloqueo se convirtieron en una oportunidad para aprender algo nuevo, para refinar la lógica del sistema o para entender mejor cómo comunicar eficazmente el frontend con el backend.

En definitiva, este proyecto ha sido mucho más que una práctica universitaria: ha sido una prueba de disciplina, resiliencia y crecimiento profesional. Multiplicar las horas dedicadas, resolver problemas reales y llegar a un resultado estable y funcional ha reforzado mi confianza como desarrollador y mi convicción de seguir aprendiendo en este camino.

1.2.4.- Aprendizaje y aplicación de nuevas tecnologías

El proyecto también ha sido una oportunidad para adentrarme en tecnologías modernas y actuales. Trabajar con Angular me permitió comprender en profundidad el desarrollo de Single Page Applications (SPA), la gestión de componentes, los servicios, la comunicación con APIs y la importancia del estado en una aplicación dinámica.

Por su parte, Django me permitió afianzar conocimientos en backend, manejo de serializadores, rutas, control de seguridad (CORS y CSRF), y gestión de contenido multimedia. Esta combinación no solo refleja el stack actual de la industria, sino que además me ha permitido aplicar metodologías profesionales dentro de un proyecto académico.

1.2.5.- Aporte al sector deportivo y a la comunidad de aficionados

El fútbol es un fenómeno global que genera una gran demanda de información, y las aplicaciones digitales son hoy su principal canal de consumo. SportsDB busca contribuir modestamente a este ecosistema, ofreciendo una plataforma centralizada donde los usuarios puedan consultar clasificaciones, resultados, estadísticas, noticias y juegos interactivos de forma sencilla.

Además de servir como práctica académica, este proyecto tiene una clara orientación al usuario final: cualquier aficionado, independientemente de su nivel de conocimiento, puede acceder a la información que le interesa sin complicaciones. Este enfoque en la accesibilidad y la experiencia de usuario es uno de los valores añadidos del trabajo.

1.2.6.- Innovación y relevancia del proyecto

Aunque existen muchas aplicaciones deportivas, SportsDB destaca por su enfoque académico, independiente y didáctico. La decisión de crear un backend propio que actúe como capa intermedia entre el usuario y las APIs externas aporta flexibilidad y control sobre la información.

Asimismo, los módulos desarrollados, como Noticias y Juego del Día, añaden una capa de personalización e interacción poco habitual en proyectos de esta índole. Con ello, SportsDB no se limita a mostrar datos, sino que construye una experiencia completa para el aficionado al fútbol, combinando información, participación y diseño intuitivo.

1.2.7.- Importancia social y cultural del fútbol digital

El fútbol ha dejado de ser un simple deporte para convertirse en un fenómeno global que mueve audiencias, economías y comunidades enteras. Las nuevas tecnologías han revolucionado la manera en que se vive este deporte: ahora los aficionados demandan inmediatez, estadísticas en tiempo real, perfiles actualizados y contenido constante.

La informática ha hecho posible que millones de personas sigan simultáneamente un mismo evento desde cualquier parte del mundo. En este contexto, proyectos como SportsDB adquieren relevancia, pues representan cómo la tecnología puede acercar la emoción del fútbol a todos los públicos, garantizando accesibilidad, rapidez y fiabilidad.

1.3.- OBJETIVOS

El proyecto SportsDB tiene como finalidad el desarrollo de una plataforma web de fútbol que centralice la información de competiciones, equipos, jugadores, resultados y estadísticas mediante la integración de APIs deportivas externas, gestionadas a través de un backend propio.

El propósito general es diseñar una aplicación útil, moderna y educativa, que permita al usuario acceder fácilmente a la información y, al mismo tiempo, al desarrollador poner en práctica los conocimientos adquiridos a lo largo del Grado en Ingeniería Informática, aplicando metodologías reales del mundo profesional.

1.3.1.- Objetivos principales

Los objetivos principales definen los logros fundamentales del proyecto, aquellos que determinan su éxito técnico y funcional.

- Diseñar y desarrollar una aplicación web completa, compuesta por un frontend en Angular y un backend en Django, comunicados mediante servicios REST.
- Integrar datos en tiempo real procedentes de APIs externas (LiveScore), encapsulándolos en endpoints propios del backend para lograr independencia del proveedor.
- Implementar un sistema de clasificación de ligas, con soporte para competiciones con y sin grupos, adaptando la estructura de datos para mantener coherencia en la interfaz.
- Mostrar resultados, partidos, jornadas y datos históricos, permitiendo navegar entre temporadas y años de competición.
- Incluir un módulo de partidos en directo, actualizando automáticamente marcadores, minutos y estado de juego.

- Diseñar una vista de equipos completa, que muestre información del club, plantilla, traspasos, estadísticas y resultados recientes.
- Incorporar la funcionalidad de “equipos favoritos”, asociando usuarios y equipos mediante una tabla de unión sin columna ID, gestionada con SQL crudo.
- Crear un módulo de noticias administrable, gestionado desde el panel de Django, con soporte para imágenes y fechas.
- Desarrollar el módulo “Juego del Día”, que permita al usuario participar en desafíos o trivias futbolísticas, fomentando la interacción.
- Asegurar la accesibilidad y usabilidad, ofreciendo una interfaz clara, fluida y responsive para dispositivos móviles, tablets y ordenadores.
- Implementar un sistema de login y registro de usuarios, con sesiones seguras, validación y protección CSRF.
- Normalizar y validar las respuestas del backend, garantizando estructuras de datos consistentes y control de errores ante fallos en la API externa.
- Servir contenido multimedia (imágenes de noticias y escudos) mediante la configuración de MEDIA_ROOT y MEDIA_URL en Django.
- Garantizar la estabilidad del sistema, manejando correctamente los errores 404, 409 y 500 en las operaciones del backend.
- Mantener la integridad de la base de datos, diseñando un modelo relacional robusto y evitando duplicados mediante claves compuestas.
- Documentar de forma exhaustiva el proceso de desarrollo, describiendo arquitectura, endpoints, diagramas y flujo de datos.
- Validar el correcto funcionamiento de la aplicación mediante pruebas de integración y casos de uso derivados de los requisitos funcionales.

1.3.2.- Objetivos secundarios

Los objetivos secundarios complementan los principales y se centran en mejorar la calidad técnica, estética y formativa del proyecto.

- Aplicar buenas prácticas de desarrollo web, asegurando código modular, reutilizable y mantenible en Angular y Django.
- Mejorar la experiencia de usuario (UX), empleando principios de diseño de interacción, colores contrastados, tipografía clara y estructuras visuales coherentes.
- Implementar navegación semántica por rutas, con URLs descriptivas para cada módulo (clasificación, equipo, noticias, juego, etc.).
- Incorporar observables (RxJS) en Angular, optimizando la gestión del estado y evitando fugas de memoria mediante async pipe.
- Optimizar el rendimiento del backend, utilizando peticiones asincrónicas, control de tiempo de respuesta y caché temporal de datos.
- Garantizar la seguridad del sistema, gestionando permisos, cabeceras CORS y validación de peticiones entre cliente y servidor.
- Implementar feedback visual de carga y errores, utilizando spinners o mensajes en pantalla para mejorar la percepción del usuario.
- Configurar el panel de administración de Django, registrando modelos y optimizando la edición de contenido desde la interfaz administrativa.
- Utilizar control de versiones (Git/GitHub) para mantener el historial del proyecto y gestionar cambios de forma estructurada.
- Aplicar Bootstrap y utilidades CSS, logrando un diseño responsive que se adapte correctamente a distintos tamaños de pantalla.
- Aprender sobre la arquitectura RESTful, profundizando en la estructura de endpoints, verbos HTTP y manejo de recursos.
- Diseñar la base de datos de forma eficiente, aplicando principios de normalización y relaciones adecuadas entre entidades.
- Implementar mensajes y respuestas JSON claros, con claves uniformes y documentación en los endpoints.
- Asegurar la escalabilidad del sistema, permitiendo futuras integraciones de nuevas ligas, deportes o fuentes de datos.

- Ofrecer un entorno de usuario personalizable, permitiendo guardar preferencias o equipos favoritos para futuras consultas.
- Desarrollar una aplicación mantenible y ampliable, que sirva como base para posteriores mejoras académicas o profesionales.
- Probar y depurar exhaustivamente, registrando errores y excepciones mediante logs detallados.
- Integrar gráficas o visualizaciones estadísticas en futuras versiones, representando datos mediante librerías o servicios externos.
- Garantizar la compatibilidad entre navegadores, verificando su funcionamiento en Chrome, Edge, Firefox y Safari.
- Promover el aprendizaje autodidacta y la documentación continua, fomentando la curiosidad por nuevas tecnologías y soluciones web.

1.4.- LÍMITES DEL PROYECTO

Aunque SportsDB ha alcanzado un desarrollo funcional completo y estable, existen ciertos límites y restricciones que acotan el alcance del proyecto dentro del marco temporal y técnico de un Trabajo Fin de Grado. Estas limitaciones no suponen carencias, sino decisiones conscientes que permiten centrar los esfuerzos en los objetivos más relevantes y realistas.

En primer lugar, una de las principales restricciones se ha debido al uso de APIs externas para la obtención de datos deportivos. Durante el desarrollo fue necesario cambiar de proveedor, pasando de la API de BeSoccer a LiveScore, lo que implicó adaptaciones en el backend y la pérdida de algunas funcionalidades específicas. Por ejemplo, ciertas secciones de información histórica como el historial completo de fichajes o el palmarés detallado de clubes no pudieron incorporarse debido a limitaciones en la información disponible o a la falta de endpoints equivalentes en la nueva API.

Asimismo, las restricciones de las propias APIs condicionan el nivel de profundidad de los datos históricos. En algunos casos, las estadísticas o clasificaciones disponibles no se extienden más allá de la temporada 2018, lo que impide ofrecer una consulta completa de registros antiguos o competiciones desaparecidas.

Otro límite identificado se relaciona con las competiciones de tipo copa. A diferencia de las ligas, estas competiciones no siguen un formato de clasificación estándar, sino un sistema de rondas eliminatorias, lo que dificulta su representación dentro de la estructura general de tablas y grupos que utiliza la aplicación. Por este motivo, las copas se muestran con una estructura simplificada respecto a las ligas regulares.

A nivel técnico, también se han definido límites en cuanto al alcance visual y de rendimiento. Aunque la interfaz es totalmente funcional y responsive, no se ha desarrollado una aplicación móvil con Android Studio (se ha hecho uso de Capacitor), ni se han implementado características más avanzadas como gráficos estadísticos, sistemas de predicción basados en inteligencia artificial o cacheo avanzado en servidor, por quedar fuera del alcance temporal del TFG. Estas funcionalidades se consideran posibles ampliaciones futuras, no requisitos de esta entrega.

Por último, y como ocurre en la mayoría de proyectos basados en fuentes de datos externas, el funcionamiento de SportsDB está parcialmente condicionado por la disponibilidad y estabilidad de los servicios de terceros. Cualquier cambio en las políticas o estructura de la API externa podría requerir futuras adaptaciones en el backend.



Capítulo 2

Antecedentes y estado de la cuestión



2.1.- SITUACIÓN ACTUAL DEL SECTOR

Hoy es difícil ver una jornada de liga sin que la mitad del estadio saque el móvil a la mínima que suena el pitido final en otro campo. El seguimiento del fútbol a través de plataformas digitales lleva años siendo algo completamente normal, pero la cantidad de opciones disponibles no ha dejado de crecer y eso ha creado un mercado bastante saturado donde las diferencias entre unas apps y otras a veces son mínimas.

El problema es que muchas de estas plataformas están pensadas para el usuario pasivo: el que entra, ve el resultado y cierra. El aficionado que quiere algo más concreto —seguir varios equipos de diferentes ligas, tener la clasificación a mano sin publicidad agresiva, consultar el historial de un equipo— acaba saltando de una app a otra para juntar la información que necesita.

A eso se suma la dependencia de APIs de pago. La mayoría de servicios gratuitos limitan bastante lo que ofrecen: solo las ligas más populares, actualizaciones lentas o funcionalidades bloqueadas hasta que pagas. Para desarrollar una aplicación propia, esto obliga a trabajar con lo que da el plan gratuito de cada proveedor y a construir una capa intermedia que gestione bien esas limitaciones.

Esa es la situación de partida de SportsDB: aprovechar las APIs disponibles para construir una plataforma propia que cubra las competiciones europeas principales, añada funcionalidades de usuario reales y tenga una interfaz que no resulte agotadora de usar.

2.2.- HERRAMIENTAS DISPONIBLES EN EL MERCADO

Para situar SportsDB respecto a lo que ya existe, se han analizado las plataformas más usadas en España para el seguimiento de resultados de fútbol

2.2.1.- Flashscore

FlashScore [33] es una de las opciones más completas del mercado. Cubre más de 30 deportes y miles de competiciones, y en fútbol concreto ofrece resultados en tiempo real, estadísticas, clasificaciones, historial de enfrentamientos y alineaciones. La actualización es casi instantánea y tiene app para Android e iOS.

El problema principal es la publicidad. La versión gratuita tiene anuncios bastante intrusivos tanto en web como en la app, y algunas funcionalidades más avanzadas están detrás de una suscripción. Para ver el resultado del partido funciona bien; para algo más, empieza a quedarse corta o cara.

2.2.2.- Sofascore

SofaScore [34] es probablemente la mejor opción gratuita para el aficionado que quiere entender el partido más allá del marcador. Ofrece mapas de calor, estadísticas de pases, duelos ganados y valoraciones en tiempo real, cosas que pocas plataformas dan sin cobrar. La interfaz está más cuidada que la mayoría y la app está bien optimizada.

Sus debilidades están en que, en competiciones secundarias los datos pueden estar incompletos, y que ciertos históricos o estadísticas específicas requieren de suscripción.

2.2.3.- LiveScore

LiveScore [35] es la opción más minimalista de las analizadas. El diseño es limpio, carga rápido y va al grano: entras, ves los resultados del día y listo. Para el usuario que solo quiere saber si ha ganado su equipo, funciona sin complicaciones.

Pero esa simplicidad también es su límite. No hay favoritos útiles, las estadísticas son básicas y no hay ninguna diferencia real entre lo que ve un usuario registrado y uno anónimo.

2.2.4.- BeSoccer

BeSoccer [36] es española y eso se nota: la cobertura de LaLiga y competiciones nacionales está por encima de cualquier competidor. Tiene datos de ligas regionales que otros ni contemplan, lo que la hace especialmente útil para el seguidor del fútbol más cercano.

Además de resultados, ofrece noticias, fichajes, estadísticas de jugadores y análisis. Es, junto a SofaScore, la más completa del mercado. Su relevancia para SportsDB es doble: es un referente como plataforma, y es también el proveedor de datos de noticias que usa SportsDB a través de su API, con las limitaciones propias del plan gratuito.

2.2.5.- 365Score

365scores [37] combina resultados con contenido editorial y permite personalizar bastante bien qué competiciones y equipos seguir. La app está bien valorada y tiene una comunidad activa. El inconveniente es que el foco tan amplio en múltiples deportes hace que el apartado de fútbol no tenga la profundidad de plataformas más especializadas. Funciona bien, pero no destaca en nada concreto.

2.2.6.- Resumen comparativo

Plataforma	Tiempo real	Estadísticas avanzadas	Favoritos	Noticias	Gratuita útil	Foco español
FlashScore	✓	Parcial	✓	✗	Limitada	✗
SofaScore	✓	✓	✓	✗	Buena	✗
LiveScore	✓	✗	Básico	✗	Buena	✗
BeSoccer	✓	✓	✓	✓	Limitada	✓
365scores	✓	Parcial	✓	✓	Buena	✗
SportsDB	✓	Parcial	✓	✓	✓	✓

Tabla 2.1. Resumen comparativo competencia

2.3.- VALORACIÓN

Las plataformas analizadas hacen bien lo que hacen. No tiene mucho sentido pretender que SportsDB viene a resolver algo que nadie más resuelve, porque no es del todo cierto. Lo que sí aporta es control. Al tener un backend propio que actúa de intermediario entre el frontend y las APIs externas, se decide exactamente qué datos llegan al usuario, cómo se normalizan y con qué lógica se presentan. Ninguna app de terceros da ese margen. También permite abstraer las inconsistencias de las APIs, algo que en la práctica ha supuesto bastante trabajo pero que hace que la aplicación sea más estable ante cambios externos.

Hay otro punto que vale la pena ser honesto sobre él: una parte importante de la motivación de este proyecto es el aprendizaje. Construir algo real con Angular, Django, Capacitor e integración de APIs externas tiene un valor técnico independientemente de que existan competidores más grandes. El hecho de que FlashScore sea mejor en cobertura no invalida lo que se aprende desarrollando un sistema similar desde cero.

Y por último, hay funcionalidades que ninguna de las plataformas analizadas ofrece: un gestor de noticias propio administrable desde el backend, un minijuego de trivia integrado o una app nativa Android construida sobre la misma base de código web. Son añadidos concretos que hacen que SportsDB sea un proyecto propio con criterios propios, no una réplica de lo que ya existe.

Capítulo 3

Hipótesis de trabajo

3.1.- TECNOLOGÍAS DE FRONTEND

3.1.1.- Angular 17

Angular [1] es el framework principal del frontend de SportsDB. Desarrollado y mantenido por Google, permite construir aplicaciones web de tipo Single Page Application (SPA) usando TypeScript como lenguaje base. La versión 17, utilizada en este proyecto, introduce los componentes standalone [2] como estándar, eliminando la necesidad de NgModules y simplificando la estructura: cada componente declara sus propias dependencias localmente.

La elección de Angular frente a React o Vue responde a un objetivo de aprendizaje explícito. Angular impone una estructura más rígida que obliga a entender conceptos como la inyección de dependencias, el enrutamiento, los observables de RxJS o la separación

entre componentes, servicios y plantillas. Esa curva de entrada más pronunciada es precisamente lo que lo hace más formativo.

Dentro del proyecto, Angular gestiona toda la capa de presentación: renderiza las vistas, gestiona la navegación entre pantallas y se comunica con el backend a través de servicios HTTP. No llama directamente a ninguna API externa; toda la lógica de integración recae en Django (ver más adelante, tecnologías backend).



Figura 3.1. Logotipo de Angular

3.1.1.1.- TypeScript

TypeScript [4] es el lenguaje en el que está escrito el frontend. Es un superconjunto de JavaScript que añade tipado estático opcional, lo que permite detectar errores en tiempo de compilación antes de que lleguen al navegador. Para alguien con experiencia previa en JavaScript, la transición es natural, pero el tipado aporta una seguridad extra especialmente útil cuando la aplicación crece en tamaño y complejidad.

3.1.1.2.- RxJS

RxJS [5] es la librería de programación reactiva que Angular utiliza para gestionar la asincronía. En SportsDB se usa principalmente a través del HttpClient de Angular para gestionar las peticiones HTTP al backend: cada llamada devuelve un Observable al que el componente se suscribe para recibir los datos cuando están disponibles. Este enfoque encaja bien con la naturaleza de la aplicación, donde la mayoría de interacciones del usuario desencadenan una petición y una actualización de la vista.

3.1.1.3.- Angular SSR

SportsDB utiliza Angular SSR (@angular/ssr) [3], el módulo de Server-Side Rendering oficial del framework. Con SSR activado, la primera carga de cada página se renderiza en el servidor antes de enviarse al cliente, lo que mejora el tiempo de carga inicial y la

indexación por motores de búsqueda. Para una aplicación de resultados deportivos donde la inmediatez importa, este aspecto tiene relevancia práctica.

3.1.2.- Bootstrap 5

Bootstrap [9] es el framework CSS utilizado para el diseño visual y la responsividad de la aplicación. Proporciona un sistema de rejilla flexible y una colección de clases utilitarias que permiten construir interfaces adaptables a diferentes tamaños de pantalla sin escribir CSS desde cero. La versión 5 elimina la dependencia de jQuery, simplificando la integración con Angular.

En SportsDB, Bootstrap gestiona la disposición general del layout (header, sidebar, contenido principal, footer) y el diseño de las tarjetas, tablas y elementos de navegación. Se combina con Angular Material en los componentes que requieren interacciones más complejas.

3.1.3.- Angular Material

Angular Material [10] es la librería de componentes UI oficial del ecosistema Angular, basada en las guías de diseño Material Design de Google. En este proyecto se usa de forma selectiva, principalmente para el CDK Overlay empleado en los modales de jugador y otros elementos de interacción que Bootstrap no cubre de forma nativa con integración Angular. No se usa como sistema de diseño principal para evitar conflictos con Bootstrap.

3.1.4.- Capacitor

Capacitor [12] es la herramienta que permite compilar la aplicación web Angular como una aplicación nativa de Android sin reescribir el código. Desarrollado por el equipo de Ionic, actúa como puente entre el código web y las APIs nativas del dispositivo móvil. En la práctica, empaqueta la aplicación dentro de un WebView y la expone como una APK instalable en Android.



Figura 3.2. Logotipo de Capacitor

La versión 7.4 utilizada genera una aplicación Android funcional que se conecta al mismo backend Django que la versión web. Esto permite mantener un único codebase para ambas

plataformas: cualquier mejora en el frontend web se refleja automáticamente en la versión móvil.

3.1.4.1.- Configuración del proyecto Android

El proyecto Android se inicializa ejecutando el siguiente comando desde la carpeta del frontend: **`npx cap add android`**.

Esto genera la carpeta **android/** con la estructura de proyecto nativa que Android Studio necesita. La configuración principal de Capacitor está en `capacitor.config.ts`, donde se define el identificador de la app (`appId`) y, en modo de desarrollo, la URL del servidor Angular para activar Live Reload:

```
// capacitor.config.ts
const config: CapacitorConfig = {
  appId: 'com.sportsdb.app',
  appName: 'SportsDB',
  webDir: 'dist/front-sports-db/browser',
  server: {
    url: 'http://192.168.18.16:4200', // Live Reload en desarrollo
    cleartext: true
  }
};
```

Código 3.1. capacitor.config.ts — configuración Live Reload Android

La IP **192.168.18.16** es la dirección del PC en la red local. Si cambia, hay que actualizarla aquí y también en **ALLOWED_HOSTS** y **CSRF_TRUSTED_ORIGINS** del `settings.py` de Django.

3.1.4.2.- Flujo de desarrollo en dispositivo físico

Durante el desarrollo se usa el modo Live Reload, que permite ver los cambios en tiempo real en el móvil sin necesidad de recompilar la APK cada vez. El flujo es:

1. Arrancar el servidor Angular accesible desde la red:

`npm start`

equivale a:

`ng serve --host 0.0.0.0 --proxy-config proxy.conf.json`

El flag **--host 0.0.0.0** hace que el servidor escuche en todas las interfaces de red, no solo en localhost, de forma que el móvil puede acceder a él por WiFi.

2. Sincronizar Capacitor y abrir Android Studio:

```
ng build
npx cap sync android
npx cap open android
```

3. Configurar el dispositivo físico: Para que Android Studio detecte el móvil hay que activar el Modo desarrollador en el dispositivo (Settings → About phone → pulsar 7 veces sobre el número de compilación) y después activar Depuración USB en las opciones de desarrollador [13]. Con el cable USB conectado y la opción de transferencia de archivos seleccionada, el dispositivo aparece en el selector de dispositivos de Android Studio.

4. Ejecutar en el dispositivo: Desde Android Studio seguir los pasos

Run → Select device → dispositivo físico → 

Con Live Reload activo, el proxy de Angular sigue gestionando las peticiones a /api/* y /user/* hacia el backend Django, de forma que el flujo de datos es idéntico al de la versión web.

Una vez hecho esto, nuestro dispositivo móvil tendrá una app de forma nativa de SportsDB, la cual podrá usar mientras estén en la misma red donde tenemos el servidor local.

De esta forma se ha realizado el desarrollo de la aplicación. Cabe objetar que si queremos montar un sistema de producción real, este procedimiento debería cambiar.

3.1.4.3.- Debug en dispositivo físico con Chrome DevTools

Una de las ventajas de las apps híbridas con Capacitor es que se pueden depurar con las mismas herramientas que se usan para el desarrollo web. Con el móvil conectado por USB y la app corriendo en el dispositivo, se abre Chrome en el PC y se navega a “chrome://inspect/#devices” [14].

Chrome detecta automáticamente el WebView de la app y muestra un enlace “inspect”. Al pulsarlo se abre un panel de DevTools completo: consola JavaScript, inspector de red, panel de elementos HTML, depurador de código TypeScript. Esto permite ver exactamente

qué peticiones se están haciendo, qué errores aparecen en consola y cómo se renderiza la interfaz en el dispositivo real, sin diferencia respecto al debug habitual en navegador de escritorio.

3.1.5.- ApexCharts

ApexCharts [11] es la librería de gráficas utilizada en el componente de estadísticas de equipo. Permite renderizar gráficos interactivos (barras, radiales, líneas) con integración nativa para Angular a través del paquete **ng-apexcharts**. Se usa concretamente para visualizar estadísticas de rendimiento como posesión, pases o remates de una forma más clara que una tabla de datos.

3.2.- TECNOLOGÍAS DE BACKEND

3.2.1.- Django 5

Django [6] es el framework web Python que actúa como backend de SportsDB. En la versión 5.2, Django expone una API REST que consume el frontend, gestiona la base de datos a través de su ORM y actúa como intermediario entre la aplicación y las APIs externas de datos deportivos.

La elección de Django frente a alternativas como Flask o FastAPI se debe a su filosofía de "baterías incluidas": viene con ORM, sistema de migraciones, panel de administración y autenticación listos para usar sin configuración adicional. Para un proyecto de esta escala, eso reduce significativamente el tiempo de configuración inicial.



Figura 3.3. Logotipo de Django

Dentro del proyecto, Django se organiza en dos aplicaciones diferenciadas. La aplicación besoccer gestiona los datos propios del sistema (usuarios, favoritos, noticias, minijuego), mientras que la aplicación livesports actúa como proxy hacia las APIs externas de datos deportivos. Esta separación facilita el mantenimiento y deja clara la responsabilidad de cada parte del código.

3.2.2.- Django REST Framework

Django REST Framework (DRF) [7] es la extensión de Django que facilita la construcción de APIs REST. Añade herramientas para serializar datos, gestionar autenticación en APIs y estructurar las respuestas HTTP de forma estandarizada. En SportsDB se usa para el manejo de respuestas JSON y la gestión de los endpoints de la aplicación besoccer. Las vistas siguen el patrón FBV (Function-Based Views) en todo el proyecto, lo que da un control más explícito sobre el flujo de cada petición.

3.2.3.- Python

Python [8] es el lenguaje de programación del backend. Su sintaxis clara y la amplia disponibilidad de librerías lo hacen una elección natural para proyectos que requieren integración con APIs externas y procesamiento de datos JSON. En este proyecto, Python se usa exclusivamente en el servidor.

3.3.- BASE DE DATOS Y ALMACENAMIENTO

3.3.1.- MySQL

MySQL [18] es el sistema gestor de base de datos relacional utilizado en SportsDB. Se ejecuta a través del servidor XAMPP en el puerto 33306 (no el estándar 3306, para evitar conflictos con otras instalaciones locales). El esquema de datos es sencillo: cinco tablas que gestionan usuarios, equipos, la relación de favoritos entre ambos, noticias y el minijuego del día.

La elección de MySQL responde a su integración natural con XAMPP, herramienta habitual en el entorno académico, y a la buena compatibilidad con Django a través del driver PyMySQL [20]. Los datos deportivos (clasificaciones, resultados, partidos) no se persisten en la base de datos; se obtienen en tiempo real desde la API externa o desde archivos de mock en desarrollo.

3.3.2.- Django ORM

El ORM de Django es la capa de abstracción que permite interactuar con la base de datos usando Python en lugar de SQL directo. En SportsDB se usa para la mayoría de

operaciones, con una excepción notable: la tabla **usuarios_equipos**, que implementa una relación muchos a muchos con clave primaria compuesta (**usuario_id**, **equipo_id**) sin columna id autoincremental.

El ORM de Django no gestiona correctamente este tipo de tablas en MySQL cuando ejecuta operaciones **INSERT**, ya que falla al intentar ejecutar **RETURNING id**. Para esos casos concretos se usa SQL crudo directamente con **django.db.connection.cursor()**. En el resto de modelos se usa ORM estándar.

3.4.- ENTORNO DE DESARROLLO Y SERVIDOR

3.4.1.- XAMPP

XAMPP [19] es el entorno de servidor local utilizado durante el desarrollo. Proporciona un servidor Apache, MySQL y phpMyAdmin en un único paquete de instalación. En este proyecto solo se utilizan MySQL y phpMyAdmin para la gestión de la base de datos. El servidor web de la aplicación lo gestiona Django directamente.

3.4.2.- Node.js y Angular CLI

Node.js [21] actúa como entorno de ejecución para las herramientas de desarrollo del frontend. Angular CLI es la interfaz de línea de comandos que permite crear componentes, ejecutar el servidor de desarrollo (ng serve), construir el bundle de producción y generar el build de Android (npm run build:android).

3.4.3.- Gestión de entornos y proxy de desarrollo

Angular gestiona configuraciones de entorno mediante archivos en src/environments/. En SportsDB hay dos:

El entorno de desarrollo web (**environment.ts**) define **production=false** y establece la URL base de la API como '/'. En este modo Angular no sabe dónde está el backend; las peticiones a **/api/*** y **/user/*** las intercepta el proxy antes de que salgan del navegador.

El entorno Android (**environment.android.ts**) define **production=true** y apunta directamente al backend con la URL completa: **http://192.168.18.16:8000/**. Aquí no hay

proxy. La app Capacitor se comunica directamente con el servidor Django en la red local, y Angular necesita la dirección completa para construir las peticiones.

No existe un tercer entorno para despliegue web. Si en el futuro la aplicación se publicara en un servidor con dominio propio, habría que crear un **environment.prod.ts** con la URL del backend en producción y registrar la configuración correspondiente en **angular.json**. El proxy de desarrollo está en **proxy.conf.json**.

```
{
  "/api": {
    "target": "http://localhost:8000",
    "secure": false,
    "changeOrigin": true
  },
  "/user": {
    "target": "http://localhost:8000",
    "secure": false,
    "changeOrigin": true
  }
}
```

Código 3.2. proxy.conf.json — configuración del proxy de desarrollo

Redirige cualquier petición que empiece por **/api/** o **/user/** al servidor Django en el puerto **8000**, de modo que el navegador nunca ve una petición **cross-origin** durante el desarrollo. El proxy está referenciado en dos sitios a la vez: en **angular.json** mediante la clave **proxyConfig**, y en el script **start** de **package.json** mediante **--proxy-config**. La duplicidad es inofensiva y refleja cómo creció el proyecto: primero se añadió en **package.json** y después se centralizó en **angular.json**.

El script de arranque también incluye **--host 0.0.0.0**, que hace que el servidor de desarrollo escuche en todas las interfaces de red. Esto permitió acceder a la aplicación desde otros dispositivos de la red local durante el desarrollo, útil para probar el comportamiento visual en móvil sin compilar la APK cada vez.

	Desarrollo web	Android
Archivo	environment.ts	environment.android.ts
production	false	true
apiUrl	/'	http://192.168.18.16:8000/
Uso	ng serve / npm start	npm run build:android

Tabla 3.1. Tabla de entornos Angular (web y Android)

3.4.4.- Django como servidor de desarrollo

En SportsDB, Django no es solo el framework del backend, también actúa como servidor durante el desarrollo. A diferencia de proyectos donde el servidor web (Apache, Nginx) se gestiona por separado, Django incluye su propio servidor de desarrollo que se arranca con un único comando:

```
python manage.py runserver 0.0.0.0:8000
```

El flag **0.0.0.0** hace que el servidor escuche en todas las interfaces de red, no solo en localhost. Esto es necesario para que la app Android en el dispositivo físico pueda conectarse al backend desde la misma red WiFi.

El puerto no estándar para MySQL. La base de datos MySQL corre en XAMPP en el puerto **33306** en lugar del estándar **3306**. Esta decisión evita conflictos con otras instalaciones de MySQL que puedan existir en el sistema. Está configurada en **settings.py** y es uno de los puntos que hay que revisar al clonar el proyecto en un nuevo equipo.

El flag **MOCK_API_MODE = True** en **settings.py** es una de las herramientas de desarrollo más útiles del proyecto. Cuando está activo, todas las llamadas a LiveScore API se sustituyen por datos cargados desde archivos JSON locales en **livesports/mocks/**. Esto permite desarrollar y probar la aplicación sin consumir la API de LiveScore, vital en desarrollo.

El panel de administración, Django Admin (**/admin**), se usa durante el desarrollo para gestionar el contenido editorial: crear y editar noticias, configurar el juego del día y administrar usuarios. Es una herramienta funcional sin necesidad de desarrollo adicional que el propio framework proporciona.

Dos servidores en paralelo. Durante el desarrollo corren simultáneamente el servidor de Angular en el puerto 4200 y el servidor de Django en el **8000**. El proxy de Angular redirige las peticiones a **/api/*** y **/user/*** hacia Django de forma transparente, como se describe en el subapartado 3.4.3. Para arrancar ambos servidores hay que abrir dos terminales:

```
# Terminal 1 - Backend
cd back-SportsDB
python manage.py runserver 0.0.0.0:8000

# Terminal 2 - Frontend
cd front-SportsDB
npm start
```

Código 3.3. Comandos de build y arranque backend frontend

3.4.5.- Git y control de versiones

Git [23] es el sistema de control de versiones utilizado durante el desarrollo de SportsDB. El repositorio está alojado en GitHub de forma privada, lo que permite mantener el historial completo del proyecto sin exponerlo públicamente.

El uso que se le ha dado es pragmático: cada vez que se completaba una funcionalidad relevante o se cerraba una fase del desarrollo — como terminar la integración de un endpoint nuevo, resolver un problema técnico importante o añadir un módulo completo, se realizaba un commit con un mensaje descriptivo del cambio.



Figura 3.4. Logotipo de Git

Este enfoque es coherente con el tamaño y naturaleza del proyecto: un único desarrollador, sin necesidad de integración continua ni trabajo en paralelo. Git ha cumplido su función esencial: tener un histórico de versiones que permite volver a cualquier punto anterior del desarrollo si algo se rompe, y documentar la evolución del proyecto a lo largo del tiempo.

3.4.6.- Visual Studio Code

El editor principal utilizado en el desarrollo ha sido Visual Studio Code [22], por su soporte nativo de TypeScript, la integración con Angular Language Service (autocompletado y detección de errores en plantillas HTML de Angular) y la gran cantidad de extensiones disponibles para Python y Django. Para la gestión de la base de datos se ha usado phpMyAdmin, incluido en XAMPP.

3.5.- APIs EXTERNAS

3.5.1.- LiveScore API

LiveScore API [15] es la pieza central de SportsDB. Sin ella, la aplicación no existiría en su forma actual: es el proveedor que suministra en tiempo real las clasificaciones,

resultados, partidos, eventos, estadísticas, alineaciones, plantillas, goleadores, historial de enfrentamientos y datos de competiciones de todas las ligas que cubre la aplicación.



Figura 3.5. Logotipo de LiveScore API

La API es un servicio independiente accesible en **livescore-api.com**, desarrollado y mantenido por Live Score API Ltd. Tiene su propio sistema de registro, credenciales y facturación, no depende de plataformas intermediarias. La autenticación se gestiona mediante una clave (key) y un secreto (secret) que se incluyen como parámetros en cada petición GET.

En SportsDB, estas credenciales nunca llegan al cliente. Se almacenan en **settings.py** del backend Django y se inyectan en cada petición desde el servidor, siguiendo el patrón BFF descrito en el subapartado 3.6.

La API organiza sus endpoints en grupos funcionales como competiciones, resultados, partidos, equipos, países, y luego dentro están los endpoints específicos como clasificación, eventos, alineaciones, etc.

Todos los endpoints devuelven JSON con la misma estructura base. Cuando **success** es **false**, la respuesta incluye un campo **error** con el motivo. En SportsDB, **api_call_with_fallback** detecta este caso y activa el **fallback** a datos de **mock** durante el desarrollo, excepto en **get_en_directo** donde **success = false** significa simplemente que no hay partidos activos en ese momento.

Limitaciones estructurales

Las dificultades que ha presentado la API no son de cantidad de datos sino de estructura: inconsistencias en los nombres de campos según el endpoint (el mismo dato puede llamarse **player_name** o **player.name** según la ruta), sistemas de paginación distintos entre endpoints (**total_pages** entero en **history** vs. URL completa **next_page** en **fixtures**), el campo **home_away** que llega como null en algunos partidos, o la mezcla de jornadas de temporadas distintas al filtrar por número de ronda. Resolver estas inconsistencias de forma centralizada en el backend fue uno de los trabajos más importantes del desarrollo, descrito en detalle en el capítulo 4.

Paginación

Dos endpoints usan sistemas de paginación distintos, lo que requirió implementar dos funciones separadas en el backend:

- En **matches/history.json** se pagina mediante un entero **total_pages** en la respuesta y el parámetro **page=N** en la petición.
- En **fixtures/list.json** se pagina mediante una URL completa **next_page** que ya incluye las credenciales.

3.5.1.1.- Postman como herramienta de exploración de la API

LiveScore API proporciona una colección oficial de Postman [24] con todos sus endpoints documentados y listos para ejecutar. Esta colección ha sido una herramienta fundamental durante el desarrollo de SportsDB para entender la estructura exacta de las respuestas antes de implementar cada endpoint en Django.

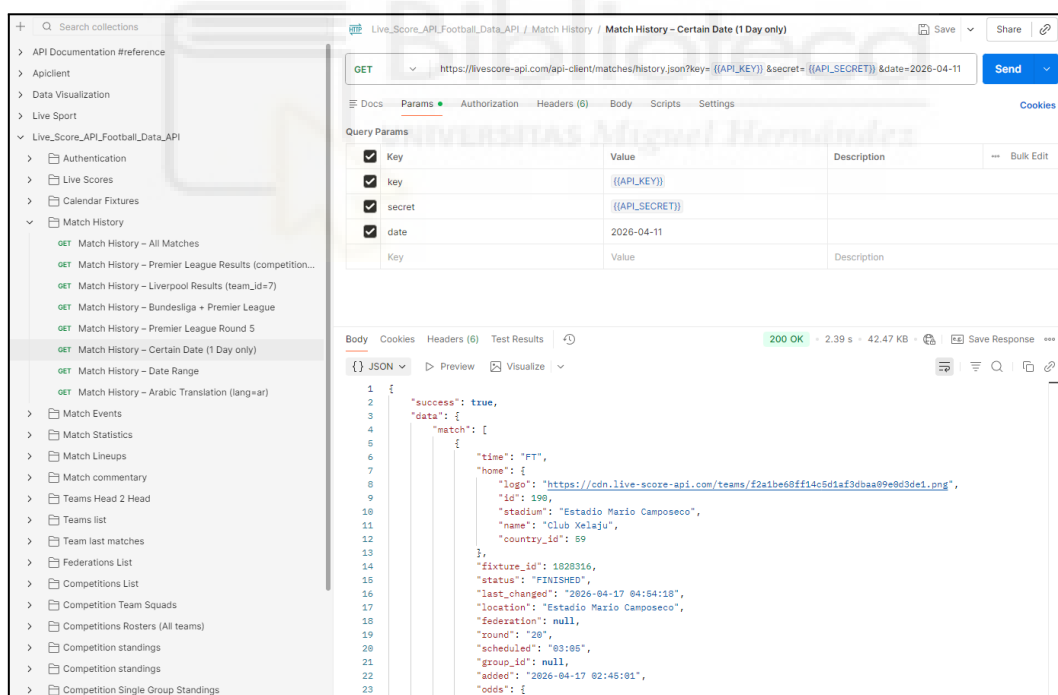


Figura 3.6. Captura de la colección de LiveScore API en Postman

Postman es una plataforma de desarrollo de APIs que permite construir, probar y documentar peticiones HTTP de forma visual, sin necesidad de escribir código. La colección de LiveScore API incluye un ejemplo de petición por cada endpoint disponible, con los parámetros necesarios y la descripción de cada campo de la respuesta.

El flujo de trabajo durante el desarrollo fue siempre el mismo: primero explorar el endpoint en Postman para entender la estructura exacta del JSON que devuelve, campos disponibles, tipos de datos, casos edge, diferencias entre competiciones, y después implementar la vista Django correspondiente. Muchas de las decisiones de normalización descritas en el capítulo 4 surgieron directamente de observar las respuestas reales en Postman: fue explorando los endpoints donde se detectó que **player_name** y **player.name** coexisten según la ruta, que **home_away** puede llegar como **null**, o que **history.json** devuelve jornadas de temporadas distintas al filtrar por número de ronda.

La colección está disponible en el repositorio del proyecto como **Live_Score_API_Football_Data_API.postman_collection.json** y se puede importar directamente en Postman añadiendo la clave y el secreto como variables de entorno.

3.5.2.- Wikipedia API

La API REST de Wikipedia [16] [17] se usa para enriquecer las fichas de equipos y jugadores con información editorial: palmarés, historia del club, descripción del jugador e imagen. Es una API pública sin autenticación que devuelve datos en formato JSON. Para jugadores con caracteres especiales en el nombre se usa como **fallback** la **Search API** de Wikipedia para localizar el artículo correcto.



Figura 3.7. Logotipo de Wikipedia

3.6.- ARQUITECTURA BFF Y PATRÓN REST

El patrón BFF (Backend For Frontend) [26] es la decisión de arquitectura más relevante del proyecto. Consiste en que el frontend nunca llama directamente a las APIs externas: todas las peticiones pasan por el backend de Django, que actúa como intermediario. Este intermediario oculta las credenciales de acceso a las APIs, transforma y normaliza los datos para que el frontend reciba siempre el mismo formato, y gestiona el **fallback** a datos de mock cuando la API no está disponible.

La alternativa habría sido que Angular llamara directamente a LiveScore API desde el navegador. Eso habría simplificado el backend pero habría expuesto las credenciales en el

cliente y habría hecho imposible el modo mock. El patrón BFF añade complejidad que en este proyecto se justifica tanto por la seguridad como por la estabilidad del desarrollo.

El flujo de una petición típica es el siguiente: el usuario navega a la sección de clasificación, el componente Angular llama al servicio **ProductsService**, este realiza un **GET** a **/api/clasificacion/{id}/**, Django recibe la petición, la delega a **livesports/views.py** que llama a LiveScore API (o al mock), transforma la respuesta y devuelve un JSON normalizado { **"status": "ok", "data": [...]** }, y Angular renderiza la tabla.

3.7.- GESTIÓN DE SEGURIDAD: CORS Y CSRF

Al separar frontend y backend en servidores distintos (puertos **4200** y **8000** en desarrollo), Django necesita configuración CORS [28][29] para permitir las peticiones cross-origin del navegador. Se usa el paquete **django-cors-headers** con **CORS_ALLOW_ALL_ORIGINS = True** en desarrollo. En un entorno de producción esta configuración debería restringirse a los dominios concretos permitidos.

Para la protección CSRF [27], Django genera un token **csrftoken** almacenado en una cookie. El frontend de Angular lee ese token manualmente y lo incluye en la cabecera **X-CSRFToken** de cada petición **POST** o **DELETE**. Una mejora pendiente es centralizar esta gestión en un interceptor HTTP global en Angular que añada el token automáticamente.

3.8.- INTELIGENCIA ARTIFICIAL COMO HERRAMIENTA DE DESARROLLO

Durante el desarrollo de SportsDB se ha usado IA generativa como apoyo. No forma parte de la aplicación, pero ha estado presente a lo largo del proyecto como herramienta de trabajo.

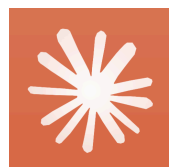


Figura 3.8. Logotipo de Claude

El uso ha sido principalmente de dos tipos. Por un lado, asistencia técnica durante la implementación: cuando había un bloqueo concreto, integrar un comportamiento

específico de Angular, resolver un error de CORS, entender cómo responde un endpoint de la API en un caso límite, se consultó al modelo Claude de Anthropic para obtener orientación. El flujo es el mismo que con cualquier otra fuente: el código que sale de ahí se revisa, se adapta y se prueba. La IA no escribe código de producción de forma autónoma; ayuda a salir de bloqueos más rápido.

Por otro lado, apoyo en la redacción de la memoria. La documentación del TFG también ha contado con asistencia de IA, principalmente para estructurar el contenido y revisar que la redacción fuera clara y coherente. El contenido técnico, decisiones de arquitectura, descripción de funcionalidades, análisis de dificultades, viene del propio desarrollo del proyecto.

Mencionar esto es lo honesto. El uso de estas herramientas es ya habitual en el sector, y omitirlo en la documentación sería impreciso.

3.9.- RESUMEN DE STACK TECNOLÓGICO

Capa	Tecnología	Versión	Función principal
Frontend web/móvil	Angular	17.3	Framework SPA principal
Lenguaje frontend	TypeScript	5.x	Tipado estático sobre JavaScript
Estilos y responsive	Bootstrap	5.3	Sistema de rejilla y utilidades CSS
Componentes UI	Angular Material	17.3	Componentes interactivos (CDK)
App nativa Android	Capacitor	7.4	Empaquetado web como APK
Gráficas	ApexCharts	5.x	Visualización estadísticas equipo
Backend	Django	5.2	Framework web Python, API Gateway
API REST	Django REST Framework	3.16	Serialización y estructura REST
Base de datos	MySQL (XAMPP)	—	Persistencia de usuarios y contenido
ORM	Django ORM	—	Abstracción SQL para modelos propios
Datos deportivos	LiveScore API	—	Clasificaciones, resultados, partidos
Enciclopedia	Wikipedia REST API	—	Fichas de equipos y jugadores
Noticias	BeSoccer	—	Feed de noticias deportivas
IDE	Visual Studio Code	—	Entorno de desarrollo principal
Servidor local	XAMPP	—	MySQL + phpMyAdmin en desarrollo

Tabla 3.2. Resumen del stack tecnológico de SportsDB

Capítulo 4

Metodología y resultados

4.1.- PLANIFICACIÓN DEL PROYECTO

4.1.1.- Ciclo de vida del software

El proyecto se ha desarrollado siguiendo un modelo de ciclo de vida incremental [30][31], en el que el sistema se construye por partes funcionales que se van añadiendo de forma progresiva. Este modelo es especialmente adecuado para proyectos donde los requisitos pueden evolucionar durante el desarrollo y donde es importante tener versiones funcionales en cada etapa.

La elección del modelo incremental frente al modelo en cascada responde a varias razones prácticas. En primer lugar, SportsDB comenzó como un proyecto de asignatura (Desarrollo de Servicios Web) con un alcance limitado, y fue creciendo orgánicamente hasta

convertirse en el TFG actual. En segundo lugar, la dependencia de APIs externas, cuyo comportamiento no siempre es predecible, hace imprescindible poder iterar rápido y ajustar sin tirar el trabajo hecho. En el siguiente Diagrama de Gantt se especifican todas las etapas del proyecto:

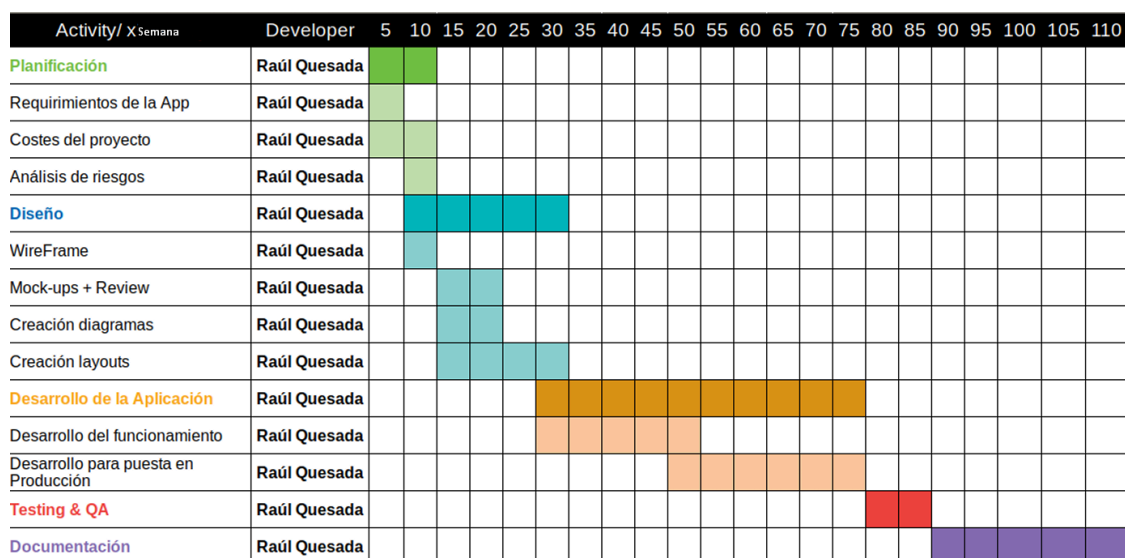


Figura 4.1. Diagrama de Gantt del proyecto

Para más información, aquí se detallan las fases que han dado lugar a toda la etapa de desarrollo:

- Fase 1: Análisis y diseño inicial: Definición de requisitos, elección del stack tecnológico, diseño del modelo de datos y mockups de las interfaces principales.
- Fase 2: Desarrollo del core: Implementación del frontend Angular con las secciones básicas (clasificación, resultados, partidos, en directo) y el backend Django con integración de LiveScore API.
- Fase 3: Gestión de usuarios: Registro, login, sistema de favoritos y panel de administración.
- Fase 4: Funcionalidades avanzadas: Detalle de partido (eventos, estadísticas, alineaciones, H2H), bracket eliminatorio, integración Wikipedia, favoritos enriquecidos.
- Fase 5: Módulos complementarios: Sección de noticias administrable desde Django Admin, minijuego de trivia.
- Fase 6: App Android y SSR: Compilación nativa con Capacitor, activación de Server-Side Rendering con Angular SSR.

- Fase 7: Documentación y pruebas: Redacción de la memoria, pruebas de funcionalidad y corrección de errores detectados.

4.2.- CAPTURA DE REQUISITOS

4.2.1.- Tipos de usuarios

Hay tres usuarios en la aplicación: Administrador, Identificado y Anónimo, siguiendo la siguiente jerarquía

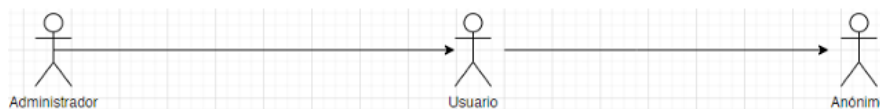


Figura 4.2. Jerarquía de tipos de usuario

Siendo de esta forma el Administrador el que más privilegios obtiene en la aplicación y el usuario anónimo es el más limitado.

Usuario	Anónimo
Descripción	No ha iniciado sesión. Puede navegar por todas las secciones públicas (clasificaciones, resultados, partidos, en directo, bracket, detalle de partido, detalle de equipo).
Casos de uso	

Tabla 4.1. Tipo de usuario anónimo

Usuario	Identificado
Descripción	Ha iniciado sesión. Dispone de todas las funcionalidades del anónimo más la gestión de equipos favoritos, noticias, juegos y posible acceso a ligas premium.
Casos de uso	

Tabla 4.2. Tipo de usuario identificado

Usuario	Administrador
Descripción	Usuario con rol admin. Tiene acceso al panel de administración de Django para gestionar usuarios, noticias y el minijuego del día.
Casos de uso	

Tabla 4.3. Tipo de usuario administrador

Los diagramas y tablas de caso de uso se recogen en el Anexo I de esta documentación al final de la misma.

4.2.2.- Requisitos funcionales

Los requisitos funcionales describen qué debe hacer el sistema. Se presentan en el formato estándar de Ingeniería del Software con identificador, descripción y condiciones de uso.

RF - 1	Visualización de la página de inicio
Actores	Todos los usuarios
Descripción	La aplicación muestra en la página principal los partidos del día agrupados por país y competición, con marcador y estado en tiempo real.

Tabla 4.4. RF-1 Visualización de la página de inicio

RF - 2	Selector de fecha en la página de inicio
Actores	Todos los usuarios
Descripción	El usuario puede navegar a días anteriores o siguientes desde la página de inicio mediante un selector de fecha.

Tabla 4.5. RF-2 Selector de fecha en la página de inicio

RF - 3	Visualización de clasificaciones
Actores	Todos los usuarios
Descripción	El usuario puede consultar la clasificación de cualquier liga disponible en el sidebar, con soporte para clasificaciones de grupos.

Tabla 4.6. RF-3 Visualización de clasificaciones

RF - 4	Visualización de clasificaciones históricas por temporada
Actores	Todos los usuarios
Descripción	El usuario puede cambiar la temporada activa en la clasificación para consultar clasificaciones de temporadas anteriores.

Tabla 4.7. RF-4 Visualización de clasificaciones históricas por temporada

RF - 5	Visualización de resultados por jornada
Actores	Todos los usuarios
Descripción	El usuario puede ver los resultados de cualquier jornada de la liga seleccionada, combinando partidos finalizados, en directo y pendientes

Tabla 4.8. RF-5 Visualización de resultados por jornada

RF - 6	Selector de jornada en resultados y partidos
Actores	Todos los usuarios
Descripción	El usuario puede navegar entre jornadas usando un selector para consultar cualquier jornada de la temporada, tanto en resultados como en partidos próximos.

Tabla 4.9. RF-6 Selector de jornada en resultados y partidos

RF - 7	Visualización de partidos próximos
Actores	Todos los usuarios
Descripción	El usuario puede consultar el calendario de la próxima jornada de la liga seleccionada, con fecha, hora y lugar de cada partido.

Tabla 4.10. RF-7 Visualización de partidos próximos

RF - 8	Partidos en directo
Actores	Todos los usuarios
Descripción	El usuario puede ver los partidos que se están disputando en ese momento con marcador actual y minuto de juego

Tabla 4.11. RF-8 Partidos en directo

RF - 9	Goleadores de una competición
Actores	Todos los usuarios
Descripción	El usuario puede consultar el ranking de máximos goleadores de la liga seleccionada.

Tabla 4.12. RF-9 Goleadores de una competición

RF - 10	Eventos de un partido
Actores	Todos los usuarios
Descripción	Al entrar en la ficha de un partido, el usuario puede consultar la línea de tiempo de eventos: goles, tarjetas, sustituciones y VAR, indicando el minuto y el jugador.

Tabla 4.13. RF-10 Eventos de un partido

RF - 11	Estadísticas de un partido
Actores	Todos los usuarios
Descripción	El usuario puede consultar las estadísticas comparadas del partido: posesión, remates, faltas, córners y tarjetas.

Tabla 4.14. RF-11 Estadísticas de un partido

RF - 12	Alineaciones de un partido
Actores	Todos los usuarios
Descripción	El usuario puede consultar los titulares y suplentes de ambos equipos con foto, dorsal y nombre.

Tabla 4.15. RF-12 Alineaciones de un partido

RF - 13	Head to Head entre dos equipos
Actores	Todos los usuarios
Descripción	El usuario puede consultar el historial de enfrentamientos directos entre los dos equipos del partido.

Tabla 4.16. RF-13 Head to Head entre dos equipos

RF - 14	Bracket eliminatorio
Actores	Todos los usuarios
Descripción	En competiciones de tipo copa, el usuario puede visualizar el cuadro eliminatorio visual con los cruces y resultados de cada ronda.

Tabla 4.17. RF-14 Bracket eliminatorio

RF - 15	Detalle de equipo
Actores	Todos los usuarios
Descripción	Al pulsar sobre cualquier equipo, el usuario accede a su ficha: plantilla, resultados recientes, próximos partidos e información general de Wikipedia (palmarés, estadio, entrenador).

Tabla 4.18. RF-15 Detalle de equipo

RF - 16	Información de jugador
Actores	Todos los usuarios
Descripción	Desde la plantilla de un equipo, el usuario puede pulsar sobre cualquier jugador para ver su ficha de Wikipedia en un modal.

Tabla 4.19. RF-16 Información de jugador

RF - 17	Registro de usuario
Actores	Usuario anónimo
Descripción	El usuario puede crear una cuenta introduciendo nickname y contraseña.

Tabla 4.20. RF-17 Registro de usuario

RF - 18	Inicio y cierre de sesión
Actores	Usuario anónimo / identificado
Descripción	El usuario puede iniciar sesión con su nickname y contraseña, y cerrarla cuando desee. La sesión persiste mediante cookies de sesión de Django.

Tabla 4.21. RF-18 Inicio y cierre de sesión

RF - 19	Gestión de equipos favoritos
Actores	Usuario identificado
Descripción	El usuario puede añadir o eliminar equipos de su lista de favoritos pulsando el icono de estrella en la ficha de equipo.

Tabla 4.22. RF-19 Gestión de equipos favoritos

RF - 20	Vista de favoritos enriquecida
Actores	Usuario identificado
Descripción	En la sección de favoritos, para cada equipo guardado se muestra el último resultado y el próximo partido en tiempo real.

Tabla 4.23. RF-20 Vista de favoritos enriquecida

RF - 21	Sección de noticias
Actores	Todos los usuarios
Descripción	La aplicación ofrece una sección de noticias deportivas con listado y vista de detalle de cada artículo.

Tabla 4.24. RF-21 Sección de noticias

RF - 22	Minijuego de trivia
Actores	Todos los usuarios
Descripción	La aplicación ofrece un minijuego diario configurable por el administrador. Las respuestas se validan en cliente sin peticiones adicionales al backend durante la partida.

Tabla 4.25. RF-22 Minijuego de trivia

RF - 23	Sidebar de navegación por ligas
Actores	Todos los usuarios
Descripción	El sidebar muestra las ligas disponibles agrupadas por país. Al pulsar una, se carga su contenido y el header muestra las pestañas correspondientes al tipo de competición.

Tabla 4.26. RF-23 Sidebar de navegación por ligas

RF - 24	Navegación con breadcrumb
Actores	Todos los usuarios
Descripción	La aplicación muestra en todo momento la ruta de navegación activa, que permite al usuario saber dónde está y volver a cualquier punto anterior con un clic. El breadcrumb recuerda el origen.

Tabla 4.27. RF-24 Navegación con breadcrumb

RF - 25	Gestión de noticias desde panel de administración
Actores	Administrador
Descripción	El administrador puede crear, editar y eliminar artículos de noticias desde el panel de Django.

Tabla 4.28. RF-25 Gestión de noticias desde panel de administración

RF - 26	Gestión del minijuego desde panel de administración
Actores	Administrador
Descripción	El administrador puede crear y configurar el juego del día desde Django Admin, indicando título, banderas y respuestas válidas.

Tabla 4.29. RF-26 Gestión del minijuego desde panel de administración

RF - 27	Gestión de usuarios desde panel de administración
Actores	Administrador
Descripción	El administrador puede consultar, modificar y eliminar usuarios desde el panel de Django.

Tabla 4.30. RF-27 Gestión de usuarios desde panel de administración

RF - 28	Aplicación Android
Actores	Todos los usuarios
Descripción	La aplicación está disponible como APK instalable en dispositivos Android, generada a partir del mismo código Angular mediante Capacitor. Ofrece las mismas funcionalidades que la versión web.

Tabla 4.31. RF-28 Aplicación Android

4.2.3.- Requisitos no funcionales

Los requisitos no funcionales describen cómo debe ser el sistema, qué cualidades de diseño debe cumplir, más allá de la funcionalidad. Se presentan en el formato estándar de Ingeniería del Software con identificador, descripción y métricas de calidad.

RNF - 1	Rendimiento
Descripción	La aplicación debe responder a las peticiones del usuario en un tiempo razonable. El detalle de partido, que combina cuatro llamadas a la API en paralelo, tiene un tiempo de respuesta objetivo por debajo de 2 segundos.

Tabla 4.32. RNF-1 Rendimiento

RNF - 2	Usabilidad
Descripción	La interfaz debe ser intuitiva y accesible para cualquier tipo de usuario. El diseño responsive permite su uso en dispositivos móviles, tablets y escritorio.

Tabla 4.33. RNF-2 Usabilidad

RNF - 3	Seguridad
Descripción	Las credenciales de las APIs externas nunca se exponen al cliente. Las peticiones que modifican datos (POST, DELETE) están protegidas por token CSRF. Las contraseñas de usuario se almacenan en la base de datos.

Tabla 4.34. RNF-3 Seguridad

RNF - 4	Disponibilidad del modo offline
Descripción	En ausencia de conexión a la API externa o en entorno de desarrollo, el sistema dispone de un modo mock (MOCK_API_MODE = True) que sirve datos desde archivos JSON locales, garantizando que la aplicación es siempre funcional durante el desarrollo.

Tabla 4.35. RNF-4 Disponibilidad del modo offline

RNF - 5	Responsabilidad
Descripción	La interfaz se adapta a diferentes resoluciones de pantalla mediante Bootstrap 5. La versión Android, generada con Capacitor, respeta las proporciones y flujos de la versión web.

Tabla 4.36. RNF-5 Responsabilidad

4.2.4.- Diagramas de casos de uso

Los diagramas de casos de uso representan de forma visual las interacciones entre los actores del sistema y las funcionalidades que este les ofrece. Siguiendo la notación UML [32], cada elipse representa un caso de uso y las líneas de asociación conectan cada actor con los casos de uso a los que tiene acceso.

SportsDB tiene tres actores diferenciados, usuario anónimo, usuario identificado y administrador, cuyas capacidades se amplían de forma incremental. El usuario identificado hereda todas las funcionalidades del anónimo y añade las propias de la gestión de cuenta y favoritos. El administrador, además de poder operar como usuario identificado, tiene acceso exclusivo al panel de Django para la gestión del contenido editorial y los usuarios del sistema.

Usuario anónimo

El usuario anónimo tiene acceso a la totalidad de las funcionalidades de consulta deportiva de la aplicación: navegación por la página de inicio (CU-1, CU-2), clasificaciones y su historial (CU-3, CU-4), resultados y jornadas (CU-5, CU-6), partidos próximos (CU-7), partidos en directo (CU-8), goleadores (CU-9), los cuatro bloques del detalle de partido (CU-10 a CU-13), bracket eliminatorio (CU-14), ficha de equipo y jugador (CU-15, CU-16), navegación por el sidebar (CU-24).

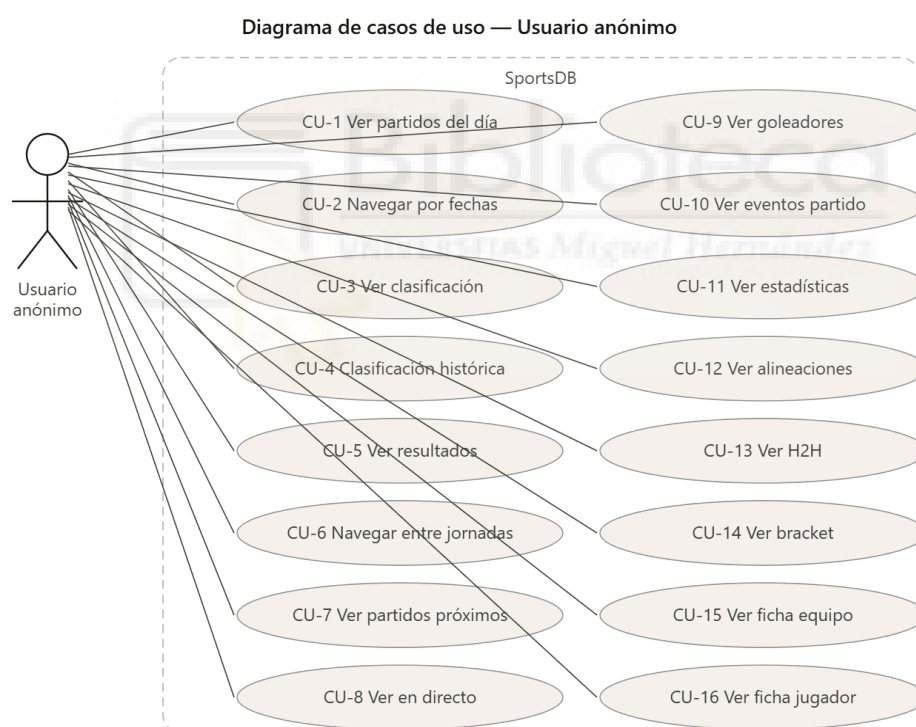


Figura 4.3. Diagrama de casos de uso — Usuario anónimo

Usuario identificado

El usuario identificado extiende al anónimo con las funcionalidades de gestión de cuenta y favoritos. Puede registrarse (CU-17), iniciar y cerrar sesión (CU-18, CU-19), añadir y eliminar equipos de su lista de favoritos (CU-20, CU-21) y consultar la vista enriquecida de sus favoritos con el último resultado y el próximo partido de cada equipo (CU-22),

noticias (CU-23) y el minijuego del día (CU-25). La aplicación Android (CU-28) también está disponible para cualquier usuario con acceso a la APK.

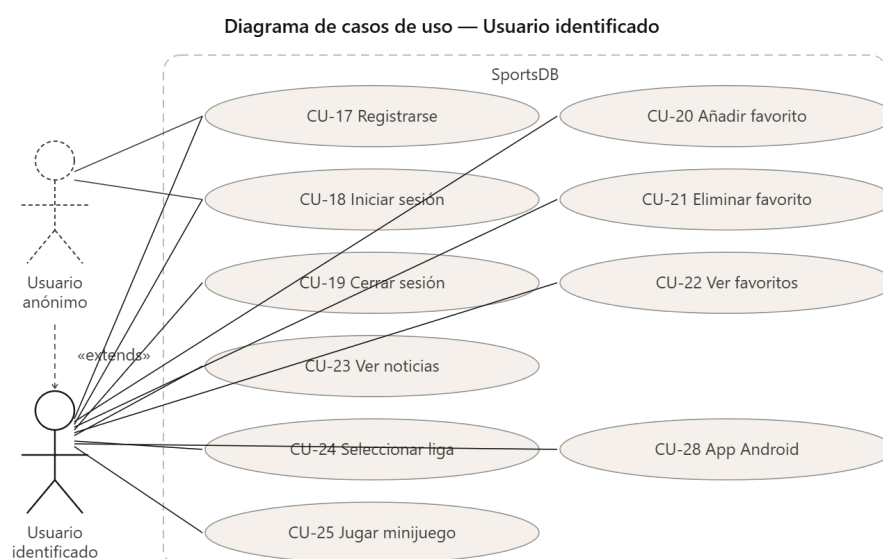


Figura 4.4. Diagrama de casos de uso — Usuario identificado

Usuario administrador

El administrador accede al panel de administración de Django mediante sus credenciales de usuario con rol admin (CU-18 como precondition, indicada con la relación «include»). Desde ese panel puede crear, editar y eliminar noticias (CU-26), configurar el minijuego del día (CU-27) y gestionar el listado de usuarios registrados (CU-28).

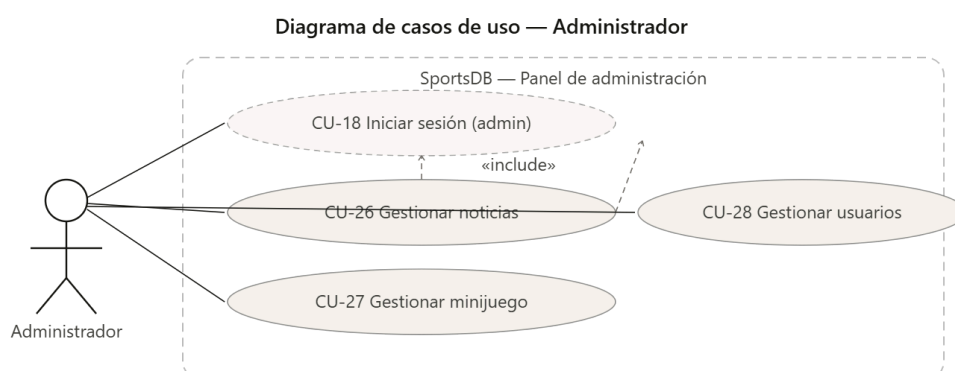


Figura 4.5. Diagrama de casos de uso — Administrador

Las tablas para una descripción completa de los casos de uso se pueden consultar en el anexo de la presente memoria

4.3.- DISEÑO

4.3.1.- Modelo de datos. Diagrama E/R

La base de datos de SportsDB es deliberadamente sencilla. Los datos deportivos (clasificaciones, resultados, partidos) no persisten, se obtienen en tiempo real desde las APIs externas. La base de datos solo almacena lo que es propiedad del sistema: usuarios, equipos guardados como favoritos, noticias editoriales y el minijuego del día. El modelo consta de cinco tablas:

- usuario: Almacena las cuentas de usuario con nickname (PK), contraseña y rol (user / admin).
- equipo: Almacena los equipos que algún usuario ha guardado como favorito. El id corresponde al identificador externo de LiveScore API.
- usuarios_equipos: Tabla de relación N:M entre usuarios y equipos (favoritos). Tiene clave primaria compuesta (usuario_id, equipo_id) sin columna id autoincremental.
- noticia: Artículos editoriales creados desde el panel de administración, con título, imagen, autor, contenido y fecha de creación automática.
- juego: Registro del minijuego del día, con título, respuestas válidas y banderas en formato JSON.

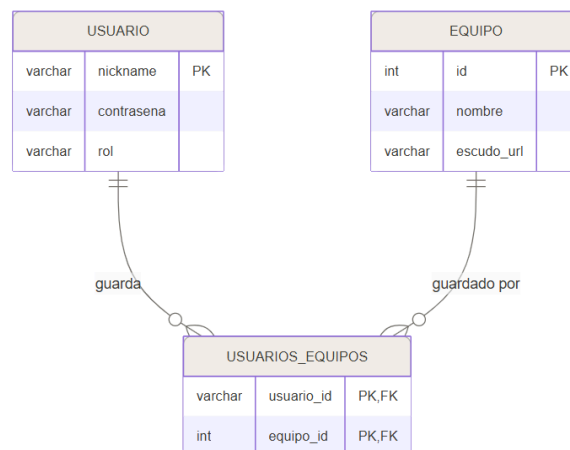


Figura 4.6 (1). Diagrama Entidad-Relación de SportsDB

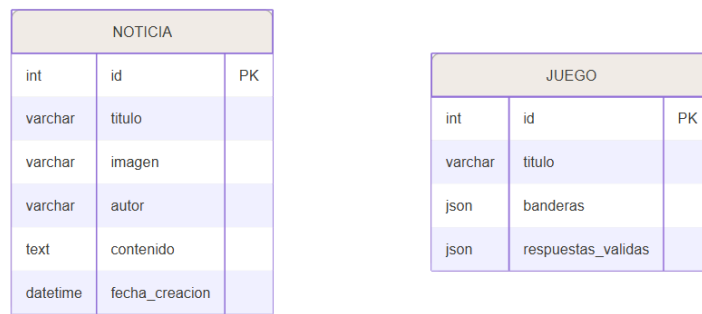


Figura 4.6 (2). Diagrama Entidad-Relación de SportsDB

La única relación explícita en el modelo es la que une usuario con equipo a través de usuarios_equipos. Las tablas noticia y juego no tienen relaciones con el resto del modelo.

4.3.2.- Arquitectura del sistema

La arquitectura de SportsDB sigue el patrón BFF (Backend For Frontend) [25], en el que el cliente Angular nunca llama directamente a las APIs externas. Todas las peticiones pasan por el backend Django, que actúa como intermediario, oculta las credenciales y normaliza los datos.

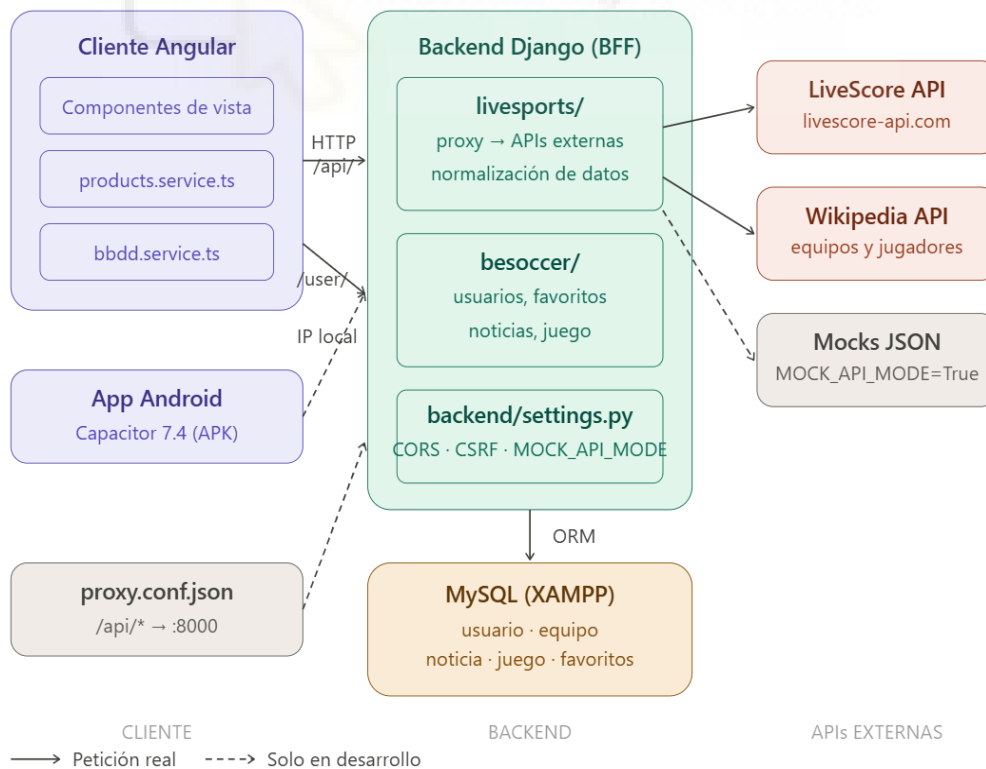


Figura 4.7. Diagrama de arquitectura BFF del sistema

4.3.3.- Diagrama de componentes angular

El frontend Angular está organizado en componentes standalone agrupados por funcionalidad. Cada componente tiene su propio template HTML, lógica TypeScript y estilos CSS encapsulados.

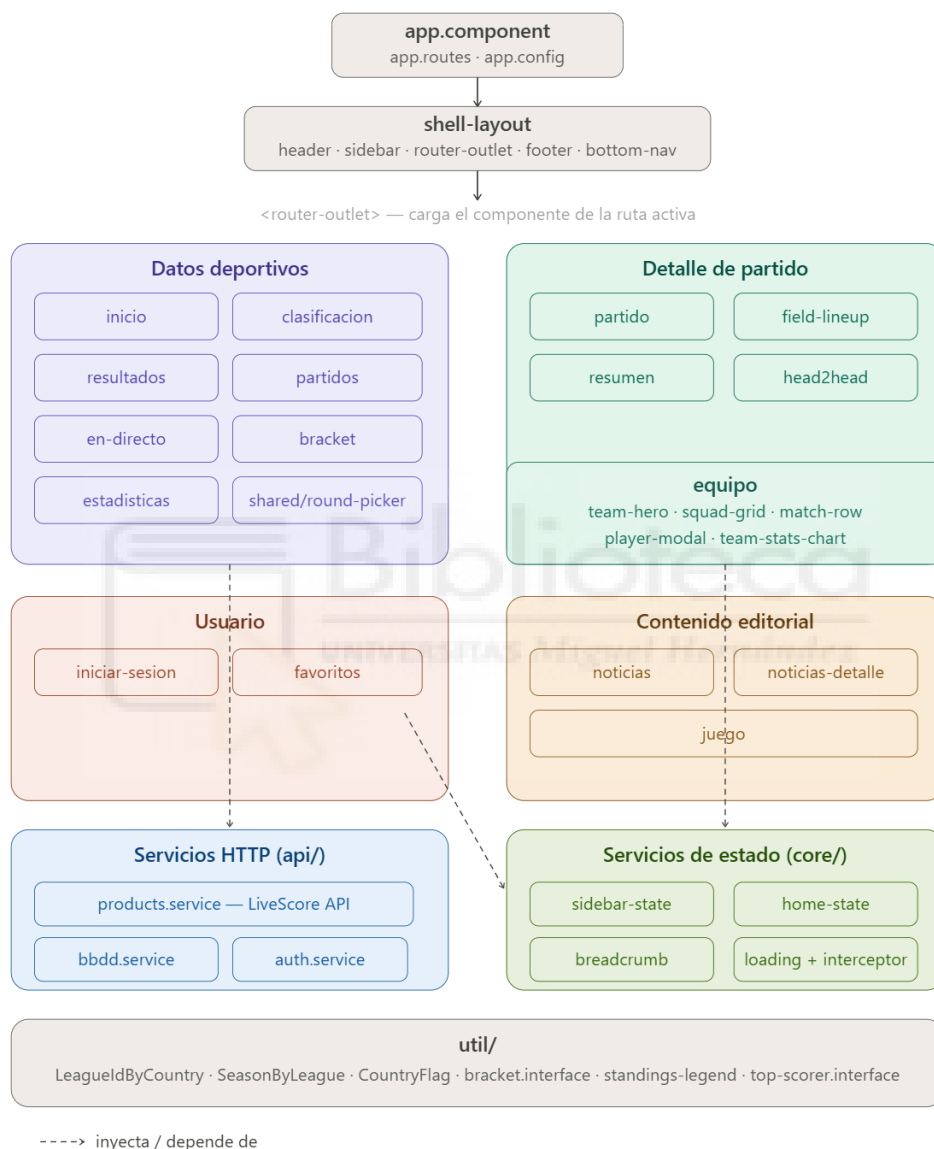


Figura 4.8. Diagrama de componentes del frontend Angular

El diagrama organiza los componentes en seis grupos con los colores codificando la categoría:

- Morado (datos deportivos): las ocho vistas principales de consulta de la API (inicio, clasificación, resultados, partidos, en directo, bracket, estadísticas) más el round-picker compartido.

- Verde azulado (detalle de partido y equipo): partido con su subcomponente field-lineup, resumen, head2head, y la ficha de equipo con sus cinco subcomponentes propios.
- Coral (usuario): login y favoritos.
- Ámbar (contenido editorial): noticias, detalle de noticia y juego.
- Azul (servicios HTTP, api/): los tres servicios que hacen las peticiones al backend.
- Verde (servicios de estado, core/): los cuatro servicios singleton más el interceptor de loading.
- Gris (utilidades, util/): enumeraciones e interfaces estáticas que no dependen de la API.
- Las flechas discontinuas muestran las dependencias de inyección: los componentes de vista inyectan los servicios HTTP y los de estado.

4.3.4.- Gestión de estado en el frontend

Una de las decisiones de arquitectura más importantes del frontend fue cómo compartir estado entre componentes que no tienen relación padre-hijo directa. Angular permite varias opciones: @Input/@Output para comunicación entre componentes relacionados, servicios singleton para estado global, y librerías como NgRx para aplicaciones con estado complejo.

En SportsDB se optó por servicios singleton sin librerías externas. NgRx añade una cantidad significativa de código repetitivo (actions, reducers, effects, selectors) que solo se justifica en aplicaciones con flujos de datos muy complejos. Para el estado que maneja esta aplicación, cuatro servicios en core/ son suficientes. Los cuatro están disponibles en toda la aplicación con providedIn: 'root':

- SidebarStateService: Gestiona el estado abierto/cerrado del sidebar. Un Signal booleano con dos métodos. El componente del sidebar y el botón hamburguesa del header lo inyectan y leen o escriben el estado directamente, sin eventos ni suscripciones:
- BreadcrumbService: Es el más complejo. Usa BehaviorSubject<Breadcrumb[]> (RxJS) que se actualiza escuchando los eventos NavigationEnd del router.

Mantiene un Map<leagueId, url> con la última URL de clasificación visitada por liga, para que al navegar a la ficha de un equipo o partido el breadcrumb sepa a qué liga volver. Ese mapa se persiste en sessionStorage para sobrevivir a una recarga de página, con helpers que envuelven el acceso en try/catch para que funcionen en SSR. Expone además setCurrentLabel() para que un componente sobrescriba el último crumb con el nombre real del equipo, que llega de la API después de que el router ya ha navegado.

- **HomeStateService:** Gestiona el estado de la pantalla de inicio: fecha seleccionada, datos de partidos, estado de carga, posición de scroll y filtro de directo. Usa Signals y computed() para el estado derivado, concretamente para aplanar y ordenar las competiciones (las más relevantes fijadas al principio según PINNED_ORDER, el resto alfabéticas).

Incluye una caché en memoria con TTL diferenciado que surgió de forma empírica. Al añadir el selector de fechas, cada navegación entre días disparaba una llamada nueva a la API aunque el usuario hubiera visto ese día diez segundos antes. La caché lo resolvió, y los TTL se razonaron a posteriori: pasado = infinito (los resultados no cambian), hoy = 5 minutos (valor intuitivo para resultados casi en tiempo real), futuro = 2 minutos (los horarios pueden cambiar por aplazamientos, caduca antes que el de hoy).

- **LoadingService:** Mantiene un contador de peticiones HTTP en vuelo con BehaviorSubject<number>. El interceptor HTTP llama a show() al iniciar cada petición y a hide() al completarla. Un debounceTime(100) evita que el overlay de carga parpadee en peticiones que resuelven muy rápido.

Servicio	Primitiva	Persistencia
SidebarStateService	Signal	Solo memoria
BreadcrumbService	BehaviorSubject (RxJS)	sessionStorage
HomeStateService	Signals + computed()	Caché en memoria con TTL
LoadingService	BehaviorSubject (RxJS)	Solo memoria

Tabla 4.37. Servicios singleton de estado en SportsDB

La elección entre Signals y BehaviorSubject responde a una diferencia práctica: los Signals son más adecuados para estado que los componentes leen directamente en el template, mientras que BehaviorSubject encaja mejor cuando hay que reaccionar a cambios de forma asíncrona, como el breadcrumb que escucha el router o el loading que necesita el debounce.

4.3.5.- Diagramas de secuencia

4.3.5.1.- Consulta clasificación

Muestra el flujo completo desde que el usuario pulsa una liga hasta que se renderiza la tabla, pasando por products.services.ts, Django, la bifurcación MOCK_API_MODE y el filtro de grupos genuinos.

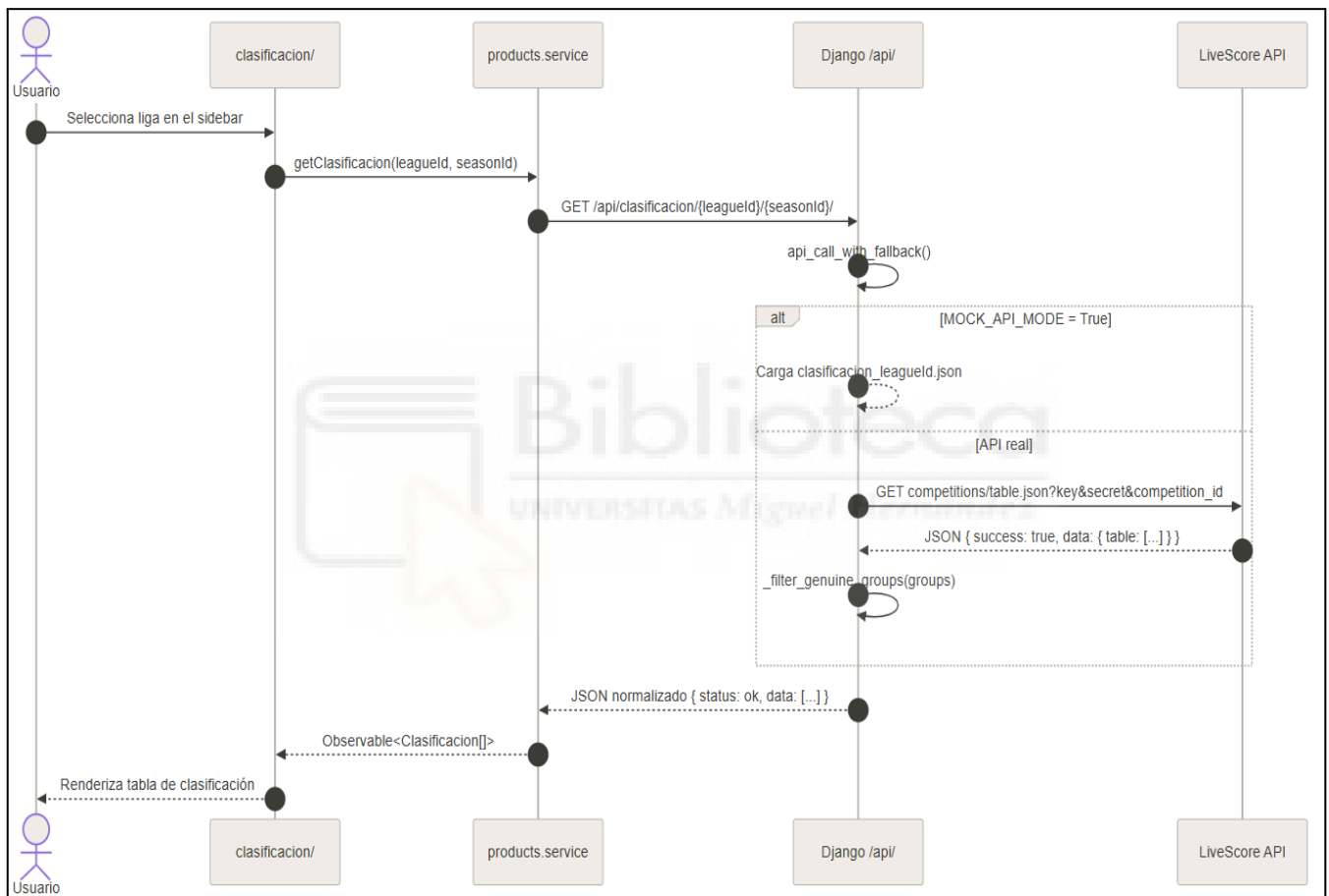


Figura 4.9. Diagrama de secuencia — Consulta de clasificación

4.3.5.2.- Añadir equipo a favoritos

Refleja los tres flujos alternativos reales del sistema: sesión no válida (401), equipo ya favorito (409) y caso de éxito con el INSERT SQL crudo por la PK compuesta. También muestra el token CSRF en cabecera y el ON DUPLICATE KEY UPDATE para insertar el equipo antes de la relación.

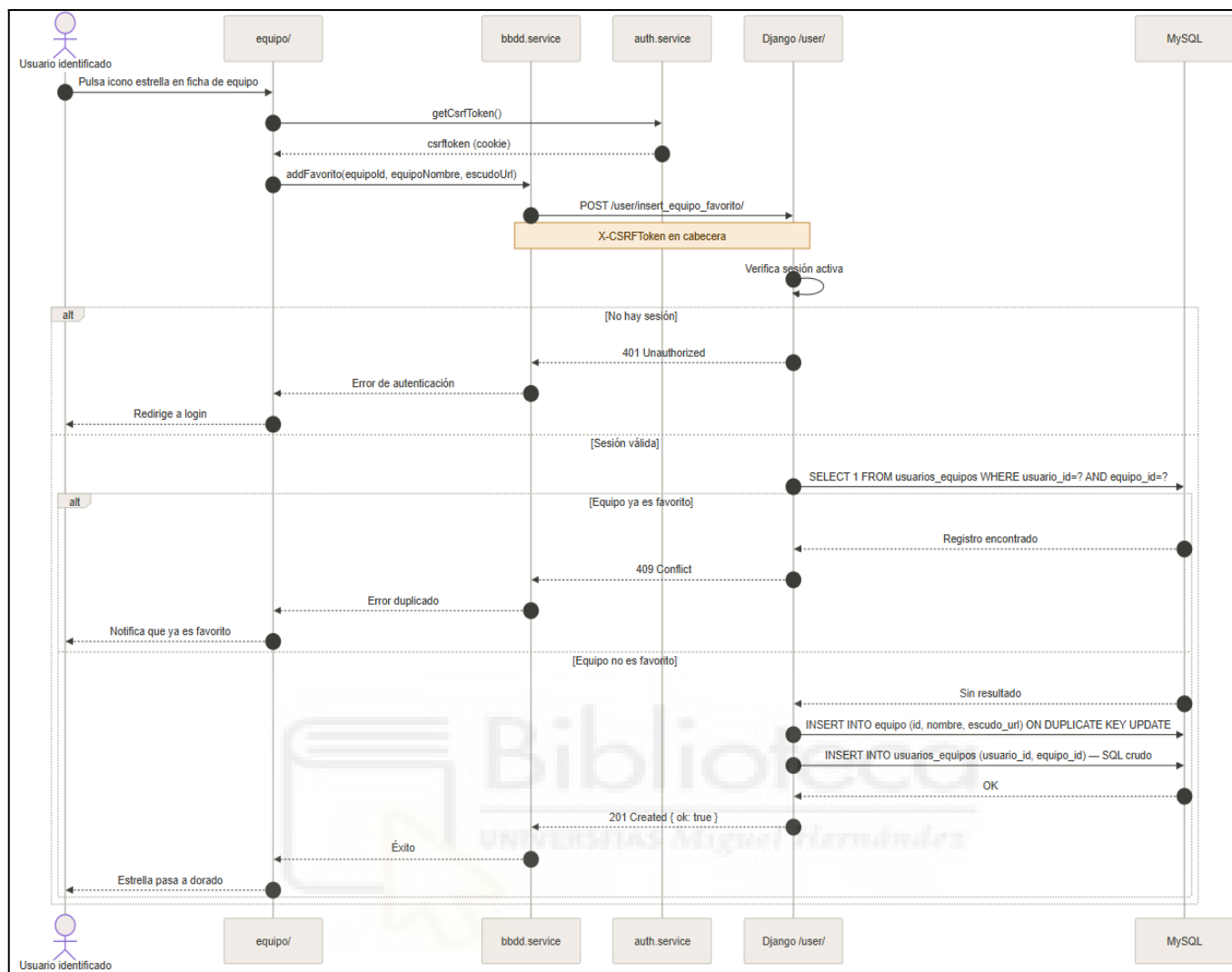


Figura 4.10. Diagrama de secuencia — Añadir equipo a favoritos

4.3.6.- Diseño de la interfaz gráfica

Antes de comenzar la implementación se diseñaron los mockups de las pantallas principales de la aplicación. Los mockups son representaciones esquemáticas de la interfaz que permiten definir la disposición de los elementos, la navegación y la jerarquía visual sin entrar en el detalle de colores ni estilos finales. Su propósito es validar el flujo de usuario y la estructura de la información antes de escribir código.

El diseño sigue en todas las vistas el mismo patrón de layout: una barra superior fija con el logo y los controles de sesión, un sidebar lateral con el listado de ligas, una cabecera de contenido con las pestañas de la liga activa, y el área principal donde se renderiza el componente de la ruta activa. En la versión móvil, el sidebar se sustituye por una barra de navegación inferior.

Página de Inicio

La pantalla de inicio es la primera que ve el usuario al abrir la aplicación. Los partidos del día se agrupan jerárquicamente por país y competición, mostrando el marcador si el partido ya ha finalizado, el minuto en rojo si está en curso, o la hora programada si aún no ha comenzado. En la parte superior del área de contenido se sitúa el selector de fecha con flechas de navegación y un botón de acceso directo a los partidos en directo.

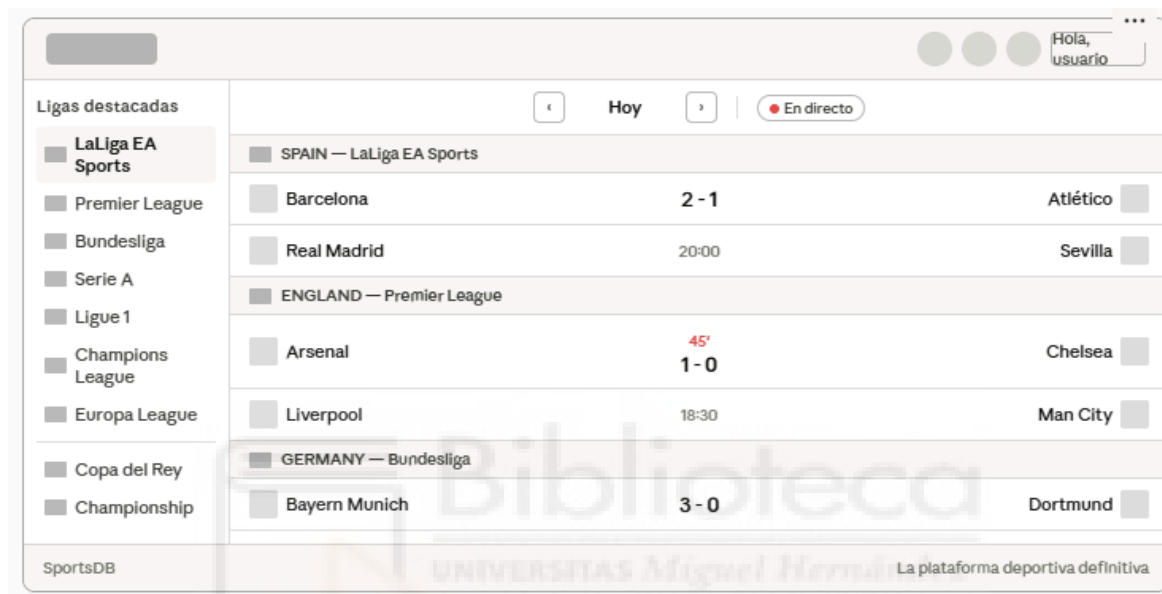


Figura 4.11. Mockup — Página de inicio

Clasificación

La pantalla de clasificación se carga automáticamente al seleccionar cualquier liga en el sidebar. Muestra la tabla con posición, escudo, nombre del equipo, puntos, partidos jugados, victorias, empates, derrotas, diferencia de goles y la forma de los últimos cinco partidos. Un selector desplegable en la esquina superior derecha permite cambiar la temporada para consultar clasificaciones históricas. Las filas se colorean según la posición: verde para el campeón y los puestos de acceso a Champions League, azul para Europa League, y rojo para los puestos de descenso. Dentro de la clasificación también se ofrecen las pestañas de Goleadores y Asistentes.

Ligas destacadas

- LaLiga EA Sports
- Premier League
- Bundesliga
- Champions League

SPAIN — LaLiga EA Sports

Clasificación Resultados Partidos En directo

Inicio · LaLiga EA Sports

Tabla de Clasificación 2025-2026 Jornada 31 · 20 equipos

Tabla Goleadores Asistentes

Pos	#	Equipo	Pts	PJ	V	E	D	DFG	Forma
1		Barcelona	79	31	26	1	4	+54	W W W D W
2		Real Madrid	70	31	22	4	5	+36	L W D W L
3		Villarreal	61	31	19	4	8	+20	W L W D W
4		Atlético Madrid	57	31	17	6	8	+19	L L W W W
5		Real Betis	46	31	11	13	7	+7	D D L D
18		Valladolid	22	31	5	7	19	-28	L L D L L

■ Campeón/UCL ■ Europa ■ Descenso

SportsDB La plataforma deportiva definitiva

Figura 4.12. Mockup — Clasificación

Resultados por jornada

La pantalla de resultados muestra los partidos de la jornada seleccionada, agrupados por fecha con separadores visuales. Cada partido muestra el escudo y nombre de local y visitante, y el estado en el centro: marcador final para partidos terminados, minuto en rojo para los que están en curso, y hora para los pendientes. Un desplegable en la parte superior permite cambiar la jornada. Al pulsar sobre cualquier partido se navega al detalle de ese encuentro.

Hola, raulqv

Ligas destacadas

- LaLiga EA Sports
- Premier League
- Bundesliga
- Champions League

SPAIN — LaLiga EA Sports

Clasificación

Resultados

Partidos

En directo

Inicio › LaLiga EA Sports › Resultados

Jornada 31 ▼

2026-04-10

Finalizado

Real Madrid

1 - 1

FINAL

Girona

2026-04-11

Finalizado

Sevilla

2 - 1

FINAL

Atlético

Barcelona

4 - 1

FINAL

Espanyol

Elche

1 - 0

FINAL

Valencia

2026-04-12

En directo

Villarreal

1 - 0

67'

Real Betis

Celta

20:00

Getafe

SportsDB

La plataforma deportiva definitiva

Figura 4.13. Mockup — Resultados por jornada

Detalle del partido

La ficha de un partido se organiza en torno a una cabecera verde oscuro con los escudos de ambos equipos, el marcador y el estado del partido. Bajo la cabecera, cuatro pestañas dan acceso a los cuatro bloques de información: Resumen (eventos), Estadísticas, Alineaciones y H2H. La pestaña de Resumen muestra la línea de tiempo de eventos en el centro del campo visual, con los eventos del equipo local a la izquierda y los del visitante a la derecha, indicando el minuto y el jugador de cada acción mediante iconos diferenciados para goles, tarjetas y cambios.



Figura 4.14. Mockup — Detalle de partido (eventos)

Ficha de equipo - Información

La ficha de equipo comparte la misma cabecera verde con escudo, nombre, estadio y forma de los últimos cinco partidos. En la esquina superior derecha, la estrella indica si el equipo está en favoritos: vacía si no lo está, dorada si lo está, y solo visible para usuarios con sesión iniciada. La pestaña de Información muestra a la izquierda el extracto introductorio de Wikipedia con un enlace directo al artículo, seguido del palmarés dividido en torneos nacionales e internacionales. A la derecha se muestra el infobox con los datos estructurados del club: nombre completo, fecha de fundación, presidente, entrenador, estadio y capacidad.



Figura 4.15. Mockup — Ficha de equipo (información Wikipedia)

Plantilla de equipo

La pestaña de Plantilla organiza los jugadores en grupos por demarcación: Porteros, Defensas, Centrocampistas y Delanteros. Cada jugador se presenta en una tarjeta con su foto, nombre y dorsal. Al pulsar sobre cualquier jugador se abre un modal con su ficha de Wikipedia.

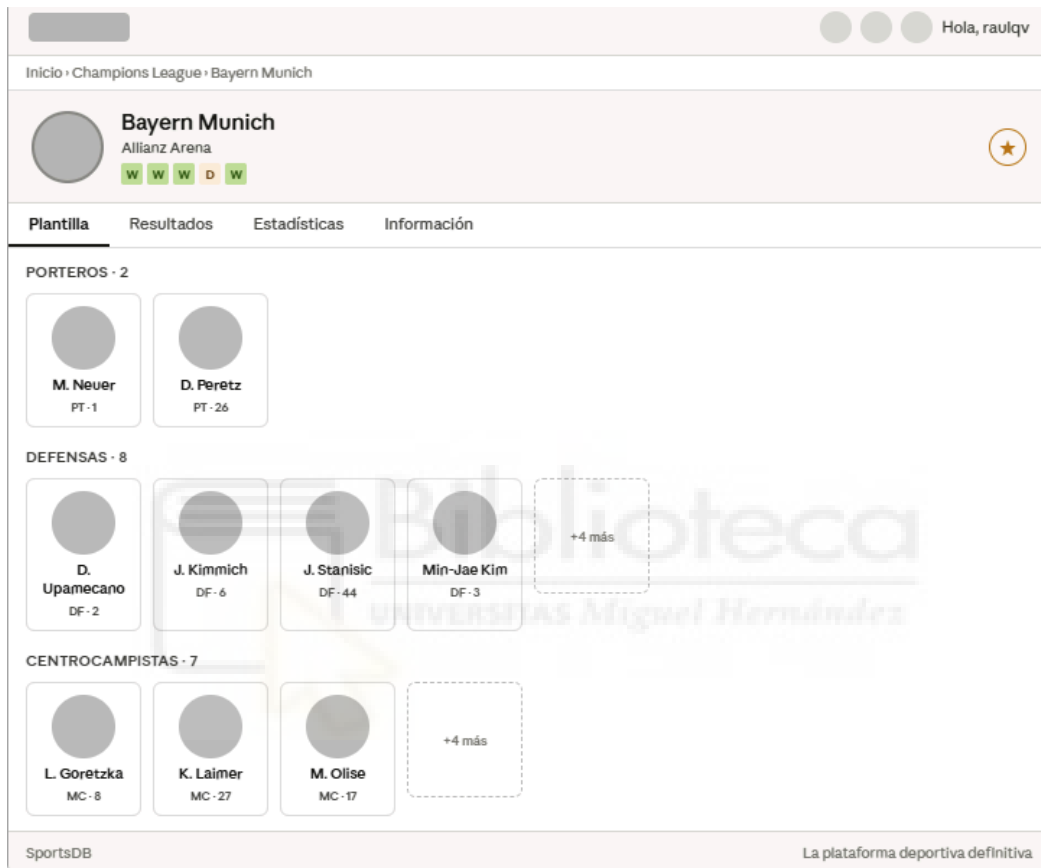


Figura 4.16. Mockup — Plantilla de equipo

Bracket eliminatorio

En las competiciones de tipo copa o mixtas, la pestaña de Eliminatorias muestra el cuadro visual con las rondas organizadas de izquierda a derecha. Cada cruce se representa como un bloque con los dos equipos y sus marcadores de ida y vuelta, con el resultado global bajo el cruce. Las rondas más avanzadas aparecen más a la derecha, siguiendo el avance natural del torneo. Los cruces pendientes muestran interrogantes en lugar de marcadores.



Figura 4.17. Mockup — Bracket eliminatorio

Favoritos enriquecidos

La pantalla de favoritos lista los equipos guardados por el usuario. Cada equipo se presenta como una tarjeta colapsable: en el estado cerrado muestra el escudo y el nombre con la estrella dorada activa; al desplegar mediante la flecha, se muestran dos bloques en paralelo con el último resultado y el próximo partido del equipo obtenidos en tiempo real. Desde cada tarjeta también se puede eliminar el equipo de favoritos.



Figura 4.18. Mockup — Favoritos enriquecidos

Login y registro

Las pantallas de autenticación prescinden del layout principal (sin sidebar ni header de ligas) para centrar la atención en el formulario. El login solicita nickname y contraseña con un enlace hacia el registro, y el registro solicita los mismos campos con un enlace de vuelta al login. Ambas pantallas comparten el mismo diseño simétrico con fondo verde oscuro y tarjeta central en blanco.

The image displays two side-by-side mobile app mockups for the SportsDB application. Both screens feature a dark green background and a central white card. The left screen, titled 'Bienvenido de nuevo', prompts the user to log in with fields for 'Nickname' and 'Contraseña', followed by an 'Iniciar Sesión' button and a link to 'Regístrate aquí'. The right screen, titled 'Crear una cuenta', prompts the user to register with fields for 'Nickname' and 'Contraseña', followed by a 'Registrarse' button and a link to 'Inicia sesión aquí'. Both screens include a 'SportsDB' logo at the bottom left.

Bienvenido de nuevo

¡Inicia sesión para continuar!

Nickname

Ingresa tu nickname

Contraseña

Ingresa tu contraseña

Iniciar Sesión

¿No tienes cuenta? [Regístrate aquí](#)

SportsDB

Crear una cuenta

¡Únete a nuestra comunidad!

Nickname

Crea un nickname

Contraseña

Crea una contraseña segura

Registrarse

¿Ya tienes cuenta? [Inicia sesión aquí](#)

SportsDB

Figura 4.19. Mockup — Login y Registro

Noticias

La sección de noticias presenta los artículos en una jerarquía visual de dos niveles: la noticia más reciente ocupa el ancho completo con imagen grande, título y autor; el resto de artículos se distribuyen en un grid de dos columnas con imagen más pequeña, título y metadatos. Al pulsar sobre cualquier artículo se navega a su vista de detalle. La sección está accesible desde el icono de noticias en el header para usuarios identificados.



Figura 4.20. Mockup — Noticias

Minijuego del día

La pantalla del juego está centrada en la vista sin sidebar, con la tarjeta del reto ocupando la mayor parte del espacio. En la parte superior de la tarjeta se muestra el título del desafío del día configurado por el administrador. Bajo el título, una barra de progreso y un contador indican cuántas respuestas se han acertado sobre el total. El campo de texto con el botón Comprobar permite introducir respuestas, y el icono de bandera permite rendirse y ver las respuestas que faltan. El grid de huecos va revelando las respuestas correctas en verde según el usuario las acierta, manteniendo en gris los huecos pendientes.

The mockup shows a web browser window with a header bar containing a logo, a menu icon, and a user greeting 'Hola, raulqv'. Below the header, a breadcrumb trail reads 'Inicio > Juego'. The main content area features a quiz card titled 'TOP 10 JUGADORES DE LA CHAMPIONS LEAGUE SEGÚN SOFASCORE'. The card includes a progress bar and a counter '3 / 10'. Below this, there is a text input field with the placeholder 'Escribe una respuesta...', a 'Comprobar' button, and a flag icon. The quiz consists of a 5x2 grid of answer boxes. The first three boxes in the first column are highlighted in green and contain the names '1. Vinicius Jr.', '2. Lamine Yamal', and '3. Harry Kane'. The remaining boxes in the first column and all boxes in the second column are grey and contain '???'.

TOP 10 JUGADORES DE LA CHAMPIONS LEAGUE SEGÚN SOFASCORE	
1. Vinicius Jr.	2. Lamine Yamal
3. Harry Kane	4. ???
5. ???	6. ???
7. ???	8. ???
9. ???	10. ???

SportsDB La plataforma deportiva definitiva

Figura 4.21. Mockup — Minijuego del día

Vistas móviles

La versión Android generada con Capacitor adapta el mismo diseño web a la pantalla del dispositivo móvil. El cambio estructural más relevante es la sustitución del sidebar lateral por una barra de navegación inferior con cuatro opciones: Inicio, Ligas, Noticias y Favoritos. El header se simplifica eliminando las pestañas de liga, que pasan a mostrarse como una fila de tabs compacta bajo la barra superior. Las tablas de clasificación reducen las columnas menos relevantes (victorias, empates, derrotas individuales) para mostrar solo posición, equipo, puntos, partidos jugados y diferencia de goles. El detalle de partido y la ficha de equipo mantienen la misma estructura de pestañas pero con tipografía y espaciado ajustados a la pantalla pequeña.

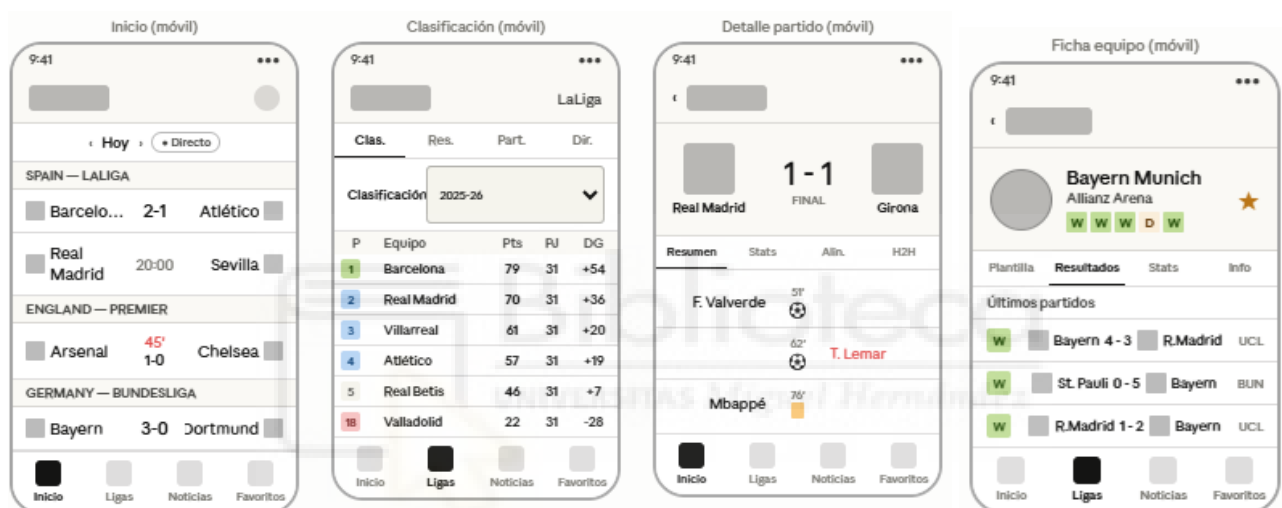


Figura 4.22. Mockup — Vistas móviles Android

4.4.- IMPLEMENTACIÓN

4.4.1.- Estructura del proyecto y sincronización entre Angular y Django

Antes de describir las funcionalidades concretas conviene entender cómo están organizados los dos proyectos que componen SportsDB y cómo se comunican entre sí. La aplicación está dividida en dos carpetas independientes dentro del mismo repositorio Git: front-SportsDB (Angular [1]) y back-SportsDB (Django [6]).

4.4.1.1.- Estructura backend Django

Django organiza el código en aplicaciones (apps), cada una responsable de un dominio funcional concreto. La configuración global del proyecto vive en la carpeta backend/, que contiene settings.py (base de datos, CORS, CSRF, apps instaladas, MOCK_API_MODE), urls.py (enrutador raíz que incluye las rutas de cada app) y los puntos de entrada WSGI/ASGI.

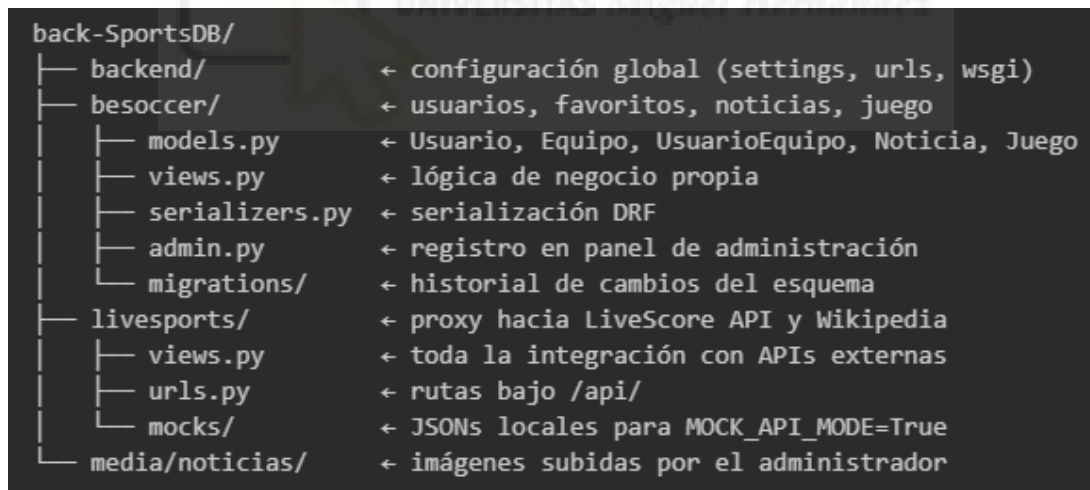


Figura 4.23. Árbol de directorios y archivos backend

SportsDB tiene dos apps diferenciadas:

- **livesports** es el proxy hacia las APIs externas. No tiene modelos propios ni persiste nada en base de datos. Su único fichero de lógica es views.py (~2100 líneas), que contiene todas las funciones de integración con LiveScore API y Wikipedia: normalización de datos, paginación, paralelización de peticiones, fallback a mocks

y agrupación de resultados. Su `urls.py` registra todas las rutas bajo el prefijo `/api/`. La carpeta `mocks/` contiene archivos JSON que replican las respuestas de cada endpoint para el modo de desarrollo offline.

- **besoccer** (nombre heredado del proveedor anterior) gestiona los datos propios del sistema: usuarios, favoritos, noticias y minijuego. Tiene modelos Django (`models.py`), serializadores DRF (`serializers.py`), vistas (`views.py`, ~670 líneas) y su propio `urls.py` con rutas bajo `/user/`. El fichero `admin.py` registra los modelos en el panel de administración de Django, desde donde el administrador gestiona noticias y el juego del día. Las `migrations/` registran la evolución del esquema de base de datos en tres pasos: creación inicial, añadir las tablas de equipos, favoritos, noticias y juego, y ajustes posteriores al modelo de juego.

4.4.1.2.- Estructura frontend Angular

Angular organiza el código en componentes, cada uno con su template HTML, lógica TypeScript y estilos CSS encapsulados. En SportsDB se usan componentes standalone (sin NgModules), lo que significa que cada componente declara sus propias dependencias directamente.

La raíz de la aplicación es `app.component.ts`, que carga el shell-layout: el contenedor principal que combina el header, el sidebar, el footer, el bottom-nav (versión móvil del sidebar) y el `<router-outlet>` donde Angular inyecta el componente de la ruta activa.

Las rutas están definidas en `app.routes.ts` y asocian cada URL a un componente. La navegación entre secciones (clasificación, resultados, partidos, detalle de partido, ficha de equipo...) no recarga la página; Angular reemplaza el contenido del `<router-outlet>` sin tocar el resto del layout.

Los servicios HTTP están en la carpeta `api/`:

- `products.services.ts` gestiona todas las llamadas a los endpoints de LiveScore API (vía Django). Es el servicio principal de la aplicación.
- `bbdd.service.ts` gestiona las llamadas a los endpoints de usuario, favoritos, noticias y minijuego.
- `auth.service.ts` gestiona el estado de autenticación: si hay sesión activa, quién es el usuario, y el token CSRF necesario para las peticiones POST y DELETE.

Los servicios de estado global están en `core/`, descritos en detalle en el subapartado 4.3.4.

Los componentes de vistas están organizados por funcionalidad: inicio/, clasificacion/, resultados/, partidos/, en-directo/, bracket/, partido/, equipo/, favoritos/, noticias/, juego/ e iniciar-sesion/. El componente equipo/ es el más complejo estructuralmente: tiene cinco subcomponentes propios (team-hero, squad-grid, match-row, player-modal y team-stats-chart).

La carpeta util/ contiene enumeraciones y utilidades estáticas que no dependen de la API: el mapeo de ligas a países (LeagueIdByCountry.enum.ts), el mapeo de temporadas por liga (SeasonByLeague.enum.ts), las ~250 banderas de países en formato PNG nombradas por código FIFA, y las interfaces TypeScript de las estructuras de datos más complejas como el bracket o los goleadores.

```
front-SportsDB/src/app/
├─ app.routes.ts           ← definición de todas las rutas
├─ app.config.ts          ← configuración de providers (HTTP, router, SSR)
├─ shell-layout/          ← contenedor principal: header + sidebar + router-outlet
├─ header/ header-content/ sidebar/ breadcrumb/ bottom-nav/ footer/
├─ api/                   ← servicios HTTP (LiveScore, bbdd, auth)
├─ core/                  ← servicios de estado global (sidebar, breadcrumb, loading, home)
├─ inicio/ clasificacion/ resultados/ partidos/ en-directo/ bracket/
├─ partido/ resumen/ head2head/
├─ equipo/                ← ficha de equipo con 5 subcomponentes
├─ favoritos/ noticias/ noticias-detalle/ juego/ iniciar-sesion/
├─ shared/round-picker/   ← selector de jornada reutilizable
├─ util/                  ← enums, interfaces y utilidades estáticas
```

Figura 4.24. Árbol de directorios y archivos frontend

4.4.1.3.- Comunicación backend-frontend

El flujo de una petición típica atraviesa varios niveles antes de llegar al dato final:

1. El usuario navega a una ruta, el router de Angular carga el componente correspondiente.
2. El componente inyecta el servicio HTTP pertinente (products.services.ts o bbdd.service.ts) y llama al método de datos que necesita.
3. El servicio construye la URL de la petición usando environment.apiUrl (que apunta a / en web o a la IP local en Android) y lanza un GET o POST con HttpClient.
4. En desarrollo web, el proxy de Angular intercepta la petición y la redirige al servidor Django en el puerto 8000.
5. Django recibe la petición en su enrutador raíz (backend/urls.py), que la delega a la app correcta según el prefijo (/api/ → livesports, /user/ → besoccer).
6. La vista Django ejecuta la lógica, llama a LiveScore API si es necesario (o carga el mock), normaliza la respuesta y devuelve un JSON al cliente.
7. Angular recibe el JSON, el servicio lo expone como Observable, el componente se suscribe y renderiza la vista.

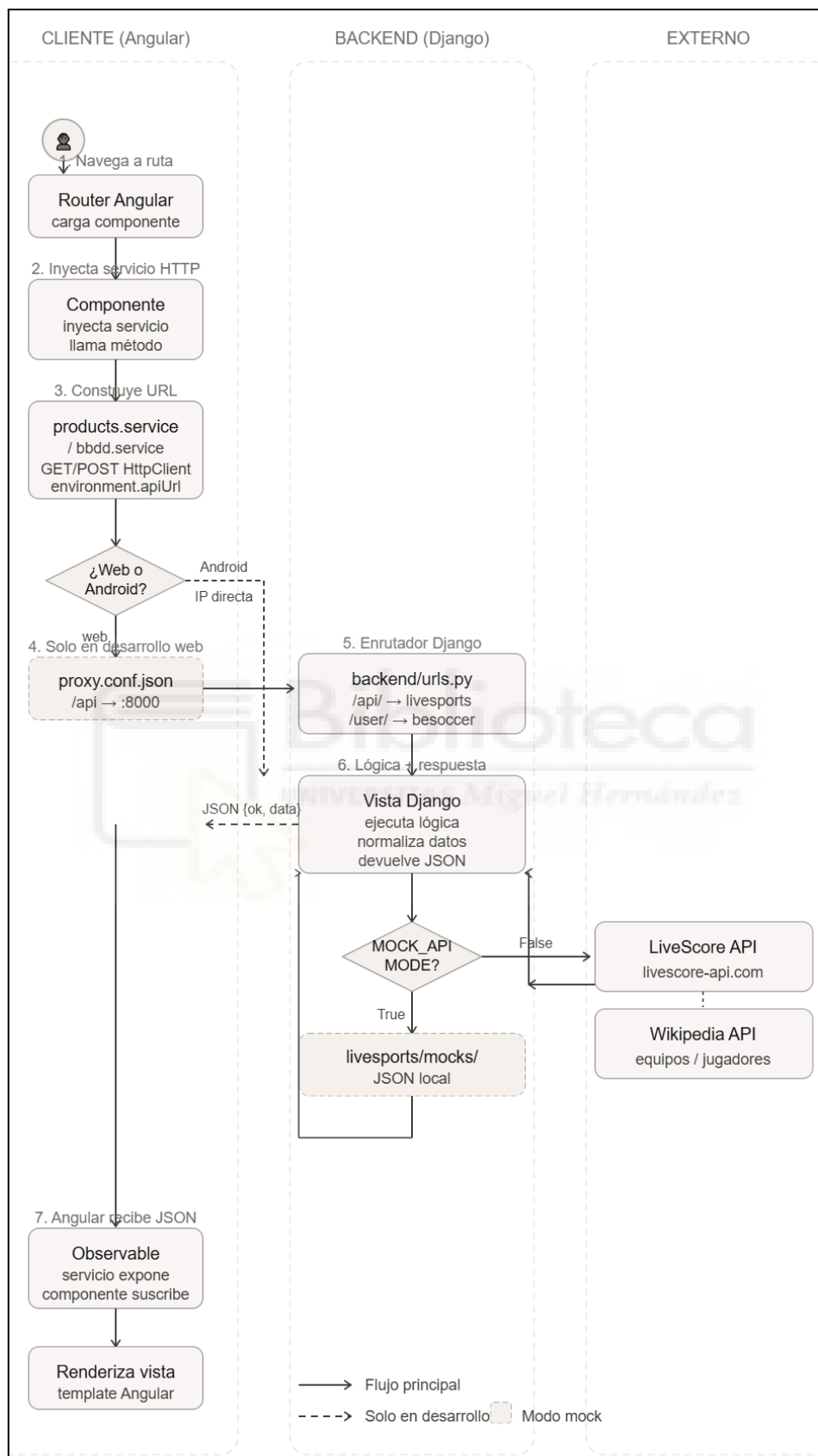


Figura 4.25. Esquema del flujo de `api_call_with_fallback`

4.4.2.- Sistema centralizado de llamadas a la API con fallback

Todas las peticiones a LiveScore API pasan por una función centralizada llamada `api_call_with_fallback`. Esta función concentra tres responsabilidades: inyectar automáticamente las credenciales de la API, gestionar el timeout y los errores de red, y cargar datos desde archivos JSON locales (mocks) cuando el modo de desarrollo offline está activado.

```
def api_call_with_fallback(endpoint_url, params, mock_filename):
    FORCE MOCK = getattr(settings, 'MOCK_API_MODE', False)

    if not FORCE MOCK:
        try:
            url = f"https://livescore-api.com/api-client/{endpoint_url}"
            if 'key' not in params:
                params['key'] = settings.LIVESCORE_KEY
            if 'secret' not in params:
                params['secret'] = settings.LIVESCORE_SECRET
            response = requests.get(url, params=params, timeout=15)
            response.raise_for_status()
            data = response.json()
            if data.get("success") is False:
                raise Exception(f"API success=false")
            return data, "API_SUCCESS"
        except Exception:
            pass

    # Fallback: cargar desde mock local
    mock_path = os.path.join(settings.BASE_DIR, 'livesports', 'mocks', mock_filename)
    with open(mock_path, 'r', encoding='utf-8') as f:
        return json.load(f), "MOCK_SUCCESS"
```

Código 4.1. `api_call_with_fallback` — sistema de fallback a mock

La función devuelve una tupla (data, source) donde source puede ser `API_SUCCESS`, `MOCK_SUCCESS` o `MOCK_FAIL`. Esto permite que cada endpoint sepa de dónde vienen los datos y actúe en consecuencia. Por ejemplo, `get_en_directo` no usa esta función porque cuando la API devuelve `success: false` no significa error, sino que no hay partidos en ese momento.

4.4.3.- Página principal: combinación de tres fuentes de datos

La página de inicio es la función más compleja del backend en cuanto a integración de datos. Muestra todos los partidos del día agrupados por país y competición, mezclando tres estados distintos: partidos ya finalizados, partidos programados y partidos en directo. Cada estado viene de un endpoint diferente de la API, con formatos de respuesta distintos.

El proceso es el siguiente:

1. Partidos finalizados (matches/history.json): se recorren todas las páginas de la API para esa fecha usando `_fetch_history_pages`, que gestiona la paginación mediante el campo `total_pages`.
2. Partidos programados (fixtures/list.json): se recorren todas las páginas usando `_fetch_all_pages`, que sigue el campo `next_page` (una URL completa) hasta que devuelve `False`.
3. Partidos en directo (matches/live.json): una sola llamada sin paginación.

El primer problema fue que la API usa dos sistemas de paginación diferentes según el endpoint. `history.json` devuelve un entero `total_pages` y espera el parámetro `page=N`. `fixtures/list.json` devuelve una URL completa en `next_page` con las credenciales ya incluidas. Esto obligó a implementar dos funciones de paginación separadas.

```
def _fetch_history_pages(base_params, max_pages=100):
    URL = "https://livescore-api.com/api-client/matches/history.json"
    items = []
    resp = requests.get(URL, params=**base_params, "page": 1, timeout=15)
    data = resp.json()
    items.extend(data.get("data", {}).get("match", []))
    total_pages = int(data.get("data", {}).get("total_pages", 1))
    for page in range(2, min(total_pages, max_pages) + 1):
        resp = requests.get(URL, params=**base_params, "page": page, timeout=15)
        items.extend(resp.json().get("data", {}).get("match", []))
    return items

def _fetch_all_pages(first_url, first_params, data_key, max_pages=50):
    items, url, params = [], first_url, first_params
    for _ in range(1, max_pages + 1):
        resp = requests.get(url, params=params, timeout=15)
        data = resp.json()
        items.extend(data.get("data", {}).get(data_key, []))
        next_page = data.get("data", {}).get("next_page")
        if not next_page:
            break
        url, params = next_page, {} # next_page ya incluye key/secret
    return items
```

Código 4.2. `_fetch_history_pages` — paginación de `history.json`

El segundo problema fue la zona horaria. El servidor de LiveScore trabaja en UTC, pero un usuario en España (UTC+2) puede estar consultando partidos de "hoy" cuando en UTC ya es "mañana", o al revés. Para cubrir este desfase de hasta dos horas, la consulta de `history` incluye el día siguiente (UTC) y la de `fixtures` incluye el día anterior.

El tercer problema fue distinguir en qué día terminó realmente un partido en directo. La API devuelve un campo date que indica la fecha de kick-off en UTC, pero un partido que empieza a las 23:00 UTC puede terminar al día siguiente. Para estos casos se usa el campo added (fecha real de finalización) en lugar de date.

Todos los partidos, independientemente de su fuente, se normalizan a un formato común mediante `_normalize_inicio_match`, que unifica las diferencias de campo entre los tres endpoints. Finalmente, `_group_inicio_matches` los agrupa jerárquicamente por país y competición.

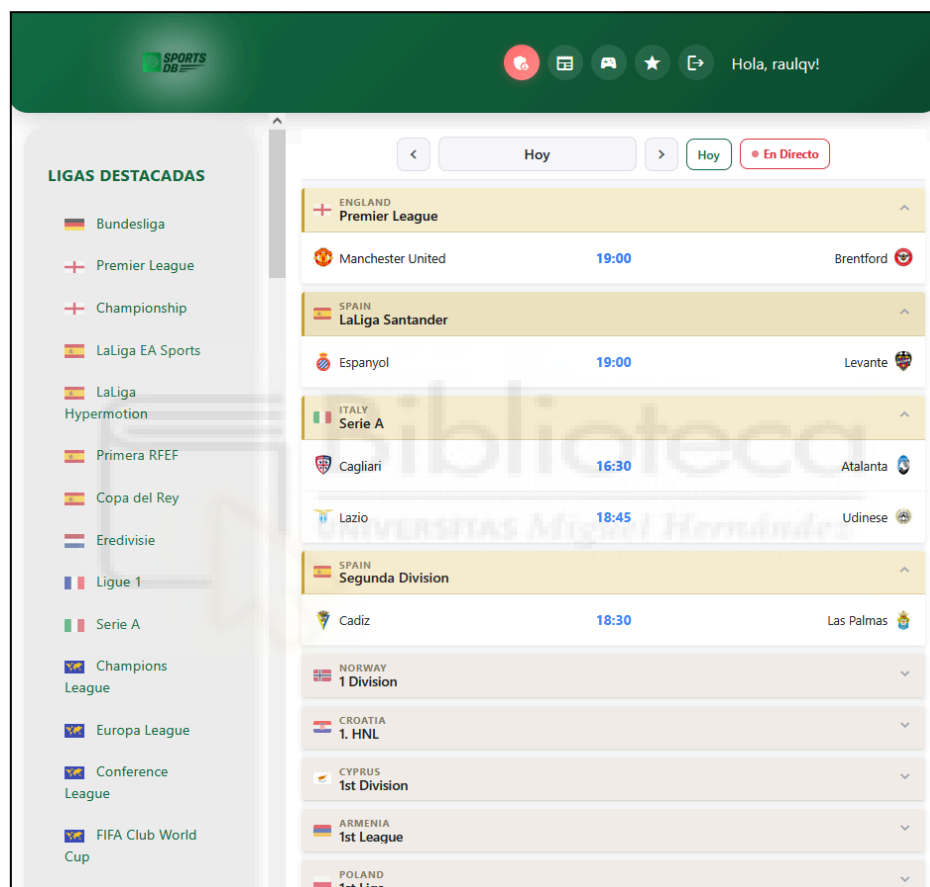


Figura 4.26. Captura de la página principal de SportsDB

4.4.4.- Resultados por jornada: fusión de tres estados de partido

La sección de resultados (`get_resultados`) necesita combinar partidos ya jugados, en directo y pendientes de una misma jornada. El problema con `history.json` es que al filtrar por número de ronda, la API puede devolver la misma jornada de temporadas anteriores. La solución fue calcular una fecha de corte (`cutoff`) basada en la fecha más reciente de los partidos pendientes o en directo, y descartar todo lo anterior a 11 meses:

```

ref_dates = [m.get("date", "") for m in pending + live_matches if m.get("date")]
ref_date = max(ref_dates) if ref_dates else str(date.today())
cutoff = ref_dt - timedelta(days=330)

played_filtered = [m for m in played if m.get("date", "") >= str(cutoff)]

```

Código 4.3. `_fetch_all_pages` — paginación de fixtures/list.json

Para los partidos en directo hubo otro problema: la API de live no siempre devuelve el campo round. La solución fue comparar con la jornada actual almacenada en caché: si la jornada solicitada coincide con la actual, se incluyen los partidos en vivo; si no, se descartan.

ESPAÑA LaLiga EA Sports		
Clasificación	Resultados	Partidos
En Directo		
Jornada 32		
2026-04-24	Finalizado	
Real Betis	1 - 1	Real Madrid
2026-04-25	Finalizado	
Atletico Madrid	3 - 2	Athletic Bilbao
2026-04-25	Finalizado	
Valencia	2 - 1	Girona
2026-04-25	Finalizado	
Getafe	0 - 2	Barcelona
2026-04-25	Finalizado	
Alaves	2 - 1	Mallorca
2026-04-26	Finalizado	
Villarreal	2 - 1	Celta Vigo
2026-04-26	Finalizado	
Osasuna	2 - 1	Sevilla
2026-04-26	Finalizado	
Real Oviedo	1 - 2	Elche
2026-04-26	Finalizado	
Rayo Vallecano	3 - 3	Real Sociedad
2026-04-27	19:00	
Espanyol	-	Levante

Figura 4.27. Captura de resultados por jornada

4.4.5.- Detalle de partido con peticiones paralelas

El detalle de partido (`get_partido_detalle`) necesita cuatro bloques de datos: eventos, estadísticas, alineaciones y H2H. Hacer estas cuatro peticiones de forma secuencial supondría unos 4-6 segundos de espera. La solución fue ejecutar las tres primeras en paralelo usando `ThreadPoolExecutor` de la librería estándar de Python, y la cuarta (H2H) a continuación, ya que necesita los IDs de los equipos extraídos de las alineaciones:

La paralelización con `ThreadPoolExecutor` reduce el tiempo total de ~5 segundos a ~1.5 segundos. Este mismo patrón se reutiliza en la función de Wikipedia para descargar palmarés e infobox en paralelo.



Figura 4.28. Captura del detalle de equipo — Wikipedia

4.4.6.- Resumen del partido: problema home/away

El endpoint de eventos de partido fue uno de los que más tiempo consumió. La API devuelve los eventos de un partido pero el campo `home_away` que indica si el evento pertenece al equipo local o visitante presenta dos problemas: a veces llega como `null` en todos los eventos, y a veces la API asigna todos los eventos al mismo lado.

La solución fue cruzar los eventos con las alineaciones para deducir el lado de cada jugador, siguiendo esta cadena de resolución:

1. Buscar el `player_id` del evento en los conjuntos de IDs de las alineaciones.
2. Si no coincide, buscar por nombre exacto.
3. Si el nombre no coincide exactamente (porque en eventos viene abreviado, por ejemplo "P. Ibañez" en lugar de "Pablo Ibañez"), intentar coincidencia por inicial + apellido.
4. Como último recurso, comparar solo el apellido.

```
def _side_by_name(name):
    nl = name.strip().lower()
    if nl in home_names: return "h"
    if nl in away_names: return "a"
    parts = nl.split()
    if len(parts) >= 2 and len(parts[0]) <= 2:
        initial = parts[0].rstrip('.')
        suffixes = parts[1:]
        for pool, side in ((home_names, "h"), (away_names, "a")):
            for full in pool:
                fp = full.split()
                if fp and fp[0].startswith(initial) and fp[-len(suffixes):] == suffixes:
                    return side
    last = parts[-1] if parts else ""
    if last:
        for pool, side in ((home_names, "h"), (away_names, "a")):
            if any(fn.split()[-1] == last for fn in pool):
                return side
    return None
```

Código 4.4. `_side_by_name` — resolución del home/away por nombre

Además, la API ha cambiado el formato del campo `home_away` a lo largo del tiempo: en versiones anteriores era "h" o "a" directamente; en versiones más recientes pasó a ser `is_home: true/false` e `is_away: true/false`. El código soporta ambos formatos.

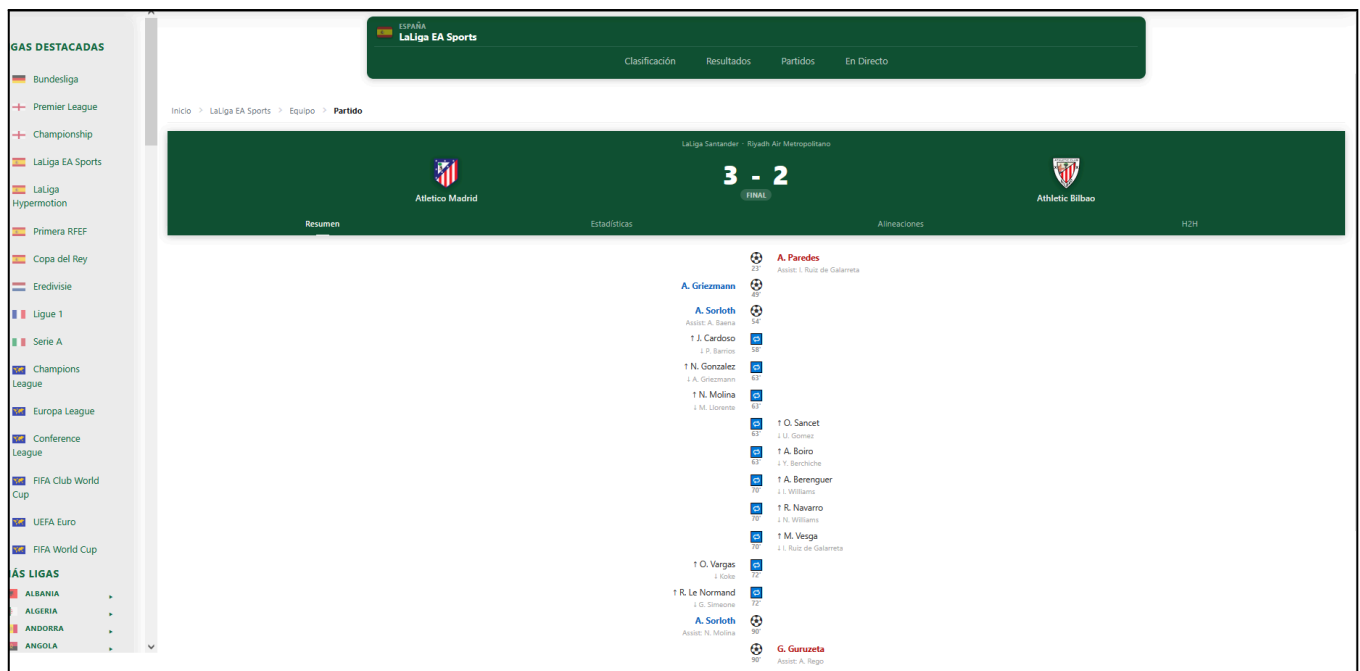


Figura 4.29. Captura del detalle de partido — tab eventos

4.4.7.- Clasificaciones y el problema de los grupos falsos

Para algunas ligas que no tienen fase de grupos real (como la Premier League), la API devuelve igualmente varios "grupos" que contienen exactamente los mismos 20 equipos repetidos. Si no se filtra, el frontend mostraría cuatro tablas idénticas.

La solución fue implementar `_filter_genuine_groups`, que recorre los grupos y solo acepta aquellos que aportan al menos un equipo nuevo:

```
def _filter_genuine_groups(groups):
    seen_teams = set()
    genuine = []
    for group in groups:
        team_ids = {entry.get("team", {}).get("id") for entry in group.get("standings", [])}
        team_ids.discard(None)
        if team_ids - seen_teams:
            genuine.append(group)
            seen_teams |= team_ids
    return genuine
```

Código 4.5. `_filter_genuine_groups` — filtro de grupos duplicados

Este filtro cubre la Premier League (4 "grupos" duplicados), ligas con grupos genuinos más grupos extra que repiten equipos, y el mundial de clubes de la FIFA donde cada grupo es completamente disjuncto.

Tabla de Clasificación												
Tabla Goleadores Asistentes												
Grupo A Grupo B Grupo C Grupo D Grupo E Grupo F Grupo G Grupo H												
Pos	#	Equipo	Puntos	PJ	V	E	D	GF	GC	DFG	Forma	
1		Palmeiras	5	3	1	2	0	4	2	2	L W D W D D	
2		Inter Miami	5	3	1	2	0	4	3	1	L D W D	
3		FC Porto	2	3	0	2	1	5	6	-1	D L D	
4		Al Ahly	2	3	0	2	1	4	6	-2	D L D W W L	

Figura 4.30. Captura de la clasificación con grupos genuinos

4.4.8.- Bracket eliminatorio y detección de fases

El bracket descarga todos los partidos históricos de una competición y los agrupa por fase. El campo round es texto libre, así que se implementó una expresión regular para distinguir automáticamente entre fases de grupos y eliminatorias:

```
_GROUP_STAGE_RE = re.compile(r"""
    ^\d+$
    | ^(\group|grupo)\s+[a-z]$
    | ^(gs|group\s+stage|fase\s+de\s+grupos)$
    | ^(matchday|md|jornada|fecha|round)\s*\d+$
    | ^(md|gw)\d+$
    """, re.IGNORECASE | re.VERBOSE)
```

Código 4.6. `_GROUP_STAGE_RE` — regex de detección de fase de grupos

Todo lo que no coincide con esos patrones se considera fase eliminatoria. Los partidos se agrupan por nombre de ronda y se invierten para mostrar el bracket de izquierda a derecha, desde octavos hasta la final.

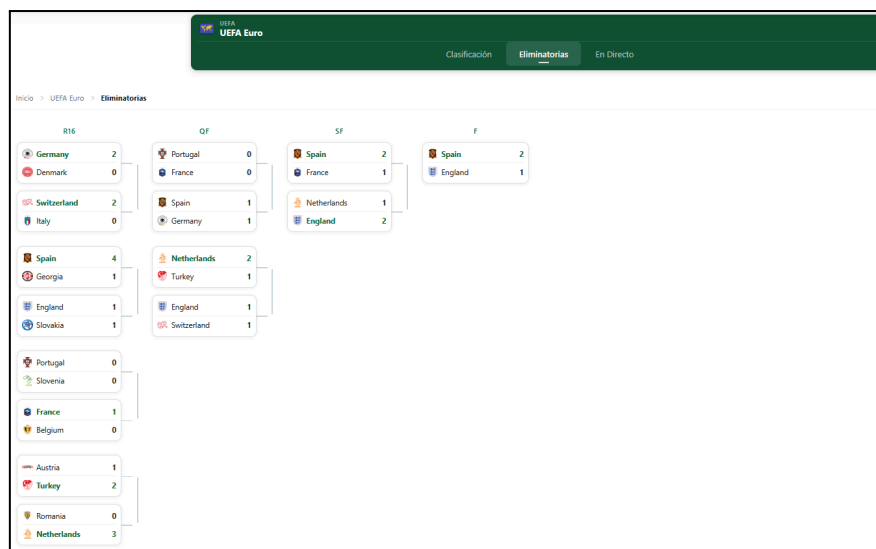


Figura 4.31. Captura del bracket de la Eurocopa

4.4.9.- Normalización de datos inconsistentes en la API

Los nombres de los campos de la API cambian según el endpoint o la competición. El backend normaliza estos datos para que Angular reciba siempre el mismo formato.

- **Goleadores:** el campo de respuesta puede llamarse scorer, topscorer o topscorers según la competición. Dentro de cada goleador, el nombre puede estar en player_name o en player.name, y la foto en player_image, player_photo, player.image o player.photo.
- **Estadísticas de partido:** la API devuelve las estadísticas como cadenas de texto con formato "55:45" (local:visitante). El backend las transforma en objetos con valores numéricos para que el frontend pueda renderizar barras de progreso directamente.
- **Plantillas de equipo:** el campo de nacionalidad puede venir como nationality o country, y la foto como image o photo. El backend unifica ambos en un formato estable.

4.4.10.- Integración de Wikipedia: scraping de palmarés e infoboxes

Para la ficha de un equipo, SportsDB extrae de Wikipedia el extracto, la imagen, el infobox (nombre, fundación, estadio, entrenador...) y el palmarés completo con el número de títulos por competición.

El nombre del equipo que viene de LiveScore no siempre coincide con el título del artículo de Wikipedia. La función `get_wikipedia_team` prueba una lista de variantes del nombre (team FC, FC team, team CF, Club de Fútbol team, etc.) y valida cada resultado comprobando que el artículo sea de fútbol mediante un conjunto de palabras clave.

El palmarés en Wikipedia viene en formato wikitext con tablas, listas, plantillas (`{{small|(3)}}`), wikilinks (`[[Copa del Rey]]`) y referencias (`<ref>`). La función `_parse_trophy_lines` procesa ambos formatos, extrae el nombre de cada trofeo y cuenta el número de títulos. Las subsecciones del palmarés se descargan en paralelo con `ThreadPoolExecutor(max_workers=5)`.





Joshua Kimmich

#6

Joshua Walter Kimmich es un futbolista alemán que juega como centrocampista o defensa en el Bayern de Múnich de la Bundesliga alemana.

DATOS PERSONALES

Nombre completo	Joshua Walter Kimmich
Nacimiento	Rottweil, Alemania · 8 de febrero de 1995 (31 años)
Nacionalidad	Alemana
Altura	1,77 m
Peso	75 kg

CLUB ACTUAL

Debut	2013 · (R. B. Leipzig)
Club	Bayern de Múnich
Posición	Defensa y centrocampista
Dorsal	6
Goles en clubes	48

SELECCIÓN NACIONAL

Selección	Alemania
Debut internacional	29 de mayo de 2016
Internacionales	108 (10)


[Ver en Wikipedia](#)

Figura 4.32. Captura del modal de jugador

4.4.11.- Sección de noticias y minijuego

Las noticias se gestionan desde el panel de administración de Django. El modelo Noticia almacena título, imagen, autor, contenido y fecha de creación. El backend expone dos

endpoints REST mediante vistas basadas en clase de Django REST Framework, las únicas CBV del proyecto.

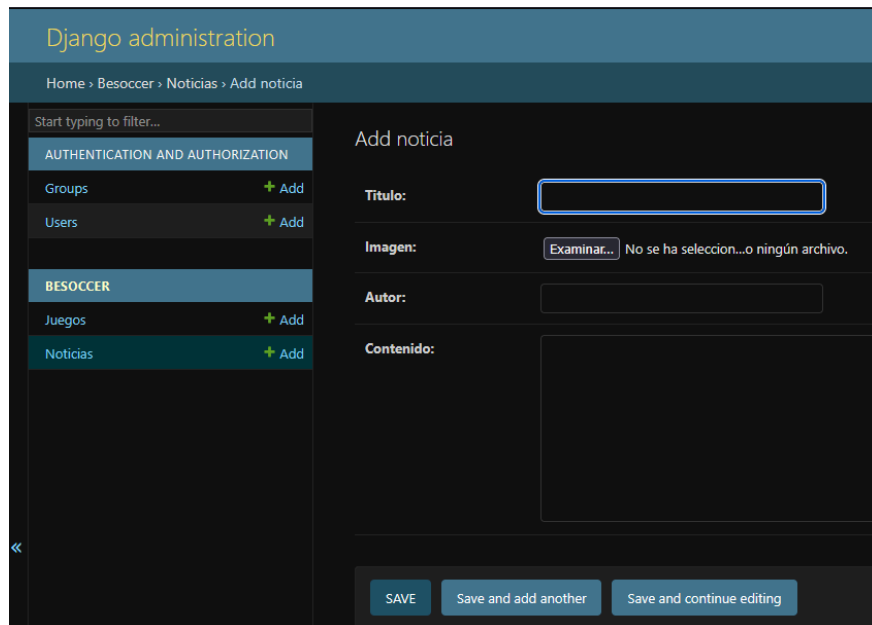


Figura 4.33. Captura del panel de administración Django

El minijuego del día funciona de forma similar: el administrador crea un registro Juego con título, las preguntas en formato JSON y las respuestas válidas. El frontend descarga el juego del día y valida las respuestas en cliente, sin peticiones adicionales al backend durante la partida.

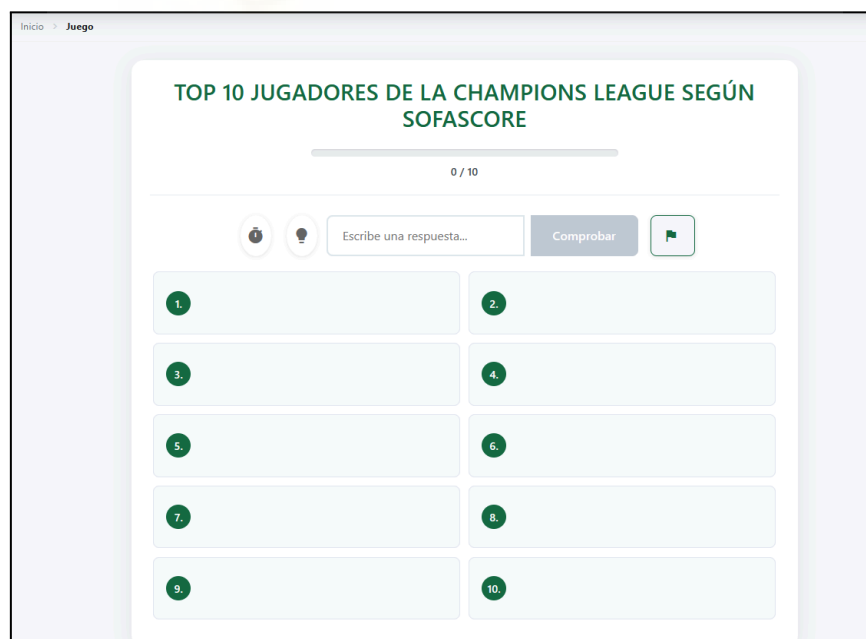


Figura 4.34. Captura del minijuego de trivia

4.4.12.- Aplicación Android con Capacitor

La versión Android de SportsDB se genera a partir del mismo código del frontend Angular usando Capacitor 7.4. El proceso de compilación es:

```
npm run build:android      # Build Angular + sincronización Capacitor
npx cap open android       # Abre Android Studio
# En Android Studio: Build → Generate Signed APK
```

Código 4.7. Compilación y arranque android

La app apunta al mismo backend Django. Para que funcione desde un dispositivo físico en la red local, la IP del servidor está configurada en environment.android.ts y en ALLOWED_HOSTS de Django.

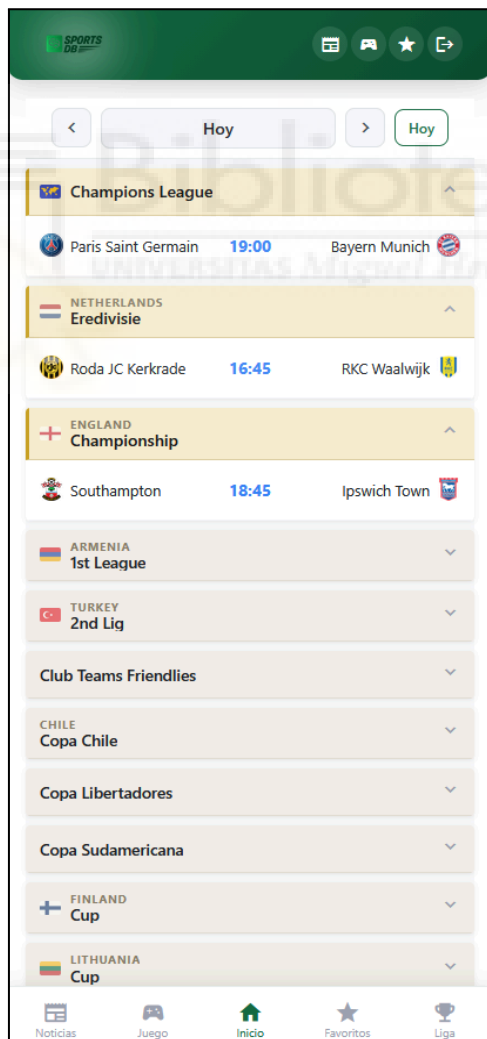


Figura 4.35. Captura de la aplicación Android en dispositivo

4.4.13.- Tipos de competición y menú dinámico

No todas las competiciones funcionan igual. Una liga pura tiene clasificación, resultados, partidos y en directo. Una copa pura solo tiene eliminatorias y en directo. Una competición mixta (Champions League) tiene clasificación por grupos, eliminatorias y en directo.

El endpoint `get_competition_info` devuelve `is_knockout` y `has_groups` para cada liga. Para las tres competiciones europeas principales (Champions, Europa League, Conference), estos valores están hardcodedos porque la API no siempre los devuelve correctamente. Para el resto, se deducen de los metadatos de la respuesta de clasificación.

4.4.14.- Caché en memoria de datos de competición

Varios endpoints necesitan saber la jornada actual de una liga, el número total de equipos, o si la competición tiene fase eliminatoria. Recalcular esto en cada petición supondría una llamada extra a la API cada vez. La solución fue mantener cuatro diccionarios en memoria a nivel de módulo (`CURRENT_ROUNDS`, `TOTAL_TEAMS`, `IS_KNOCKOUT`, `HAS_GROUPS`) que se actualizan automáticamente cada vez que se consulta la clasificación de una liga.

Este sistema no es perfecto: si el servidor se reinicia, la caché se pierde. Pero para un proyecto que corre en entorno local es una solución razonable que ahorra llamadas y reduce la latencia.

4.4.15.- Dificultades del frontend y soluciones adoptadas

Esta sección recoge los problemas más significativos que surgieron durante el desarrollo del frontend y cómo se resolvieron.

- La página de inicio no existía: La primera versión de la pantalla de inicio mostraba la clasificación de una liga hardcodeda. No había una pantalla real con todos los partidos del día. El problema era conceptual: ¿cómo obtener partidos de todas las competiciones en un único endpoint? La solución llegó al entender que la API tiene tres endpoints complementarios: `matches/live.json`, `matches/history.json` y `fixtures/list.json`. Combinando los tres y filtrando por fecha se puede construir una pantalla de inicio coherente con todos los partidos del día de todas las ligas, agrupados por país y competición.

- La distinción resultados / partidos: Durante bastante tiempo los componentes de resultados y partidos próximos estaban mal diferenciados. La definición final es simple pero costó llegar a ella: "resultados" son los partidos ya jugados de una jornada (history) y "partidos" son los que quedan por jugar (fixtures).
- El sidebar y app-header-content: El sidebar y el componente app-header-content no tienen relación padre-hijo directa con los componentes de contenido. La solución fueron los servicios singleton descritos en 4.3.4. El app-header-content tuvo además su propio reto: el menú activo no siempre se actualizaba correctamente porque Angular reutiliza los componentes al navegar entre rutas del mismo componente. La solución fue suscribirse a los eventos del router para forzar la actualización del estado activo en cada navegación.
- El breadcrumb: Sin breadcrumb la aplicación se sentía un laberinto. El BreadcrumbService resolvió esto generando automáticamente la ruta de migas de pan en cada navegación y recordando la liga de origen para poder volver a ella desde cualquier punto.
- La vista móvil y el bottom-nav: Adaptar la interfaz a pantallas pequeñas fue uno de los aspectos que más iteraciones requirió. La solución fue añadir un bottom-nav (barra inferior) que reemplaza al sidebar en móvil, siguiendo el patrón habitual de las apps Android e iOS.
- Sin buscador por restricción de la API: SportsDB no tiene buscador global. No es una decisión de diseño ni algo pendiente por tiempo: LiveScore API no expone ningún endpoint de búsqueda. Implementarlo habría requerido descargar y cachear en local el catálogo completo o contratar un plan superior.
- El bracket visual: El objetivo era un cuadro eliminatorio visual similar a los torneos de tenis. Las limitaciones de la API lo complicaron: los datos vienen como lista de partidos, no como árbol, y el nombre de cada ronda es texto libre que varía según la competición. El resultado final funciona bien para las competiciones europeas principales, donde las rondas están bien definidas.
- Lógica de acceso contextual: Una decisión que costó definir fue dónde se puede acceder a cada tipo de información. La regla final: todo lo referente a un partido solo es accesible desde partidos y resultados. Las fichas de jugador solo se ven desde la plantilla de un equipo.
- ThreadPoolExecutor en lugar de asyncio: Para las peticiones paralelas en el detalle de partido y en el palmarés de Wikipedia, la primera opción considerada fue asyncio con aiohttp. Pero Django con WSGI es síncrono: las vistas son funciones

síncronas y el servidor no gestiona un bucle de eventos asíncrono. La decisión fue pragmática: ThreadPoolExecutor de la librería estándar funciona dentro de vistas síncronas, paraleliza llamadas I/O de red sin tocar la arquitectura, y reduce el tiempo de respuesta del detalle de partido de ~5 segundos a ~1.5 segundos.

4.4.16.- Problemas de compatibilidad con Angular SSR

Activar SSR trajo consigo una serie de problemas que no aparecen en aplicaciones Angular sin él. Durante el renderizado en servidor no existe el objeto window, ni document, ni localStorage. Cualquier código que acceda a estas APIs del navegador falla en el servidor con errores como ReferenceError: window is not defined.

La siguiente tabla resume los casos concretos que surgieron y cómo se resolvieron:

Problema	Solución aplicada	Resultado
document.cookie (lectura del token CSRF)	isPlatformBrowser + inject(DOCUMENT)	Correcto
ApexCharts y otras librerías DOM	isPlatformBrowser con flag + early return	Correcto
localStorage en logout	typeof window !== 'undefined'	Funciona, no idiomático
localStorage en favoritos/login	Sin guard	Latente, no explota por flujo de navegación
window.scrollTo	Sin guard	Latente, no explota por ser en eventos de click

Tabla 4.38. Compatibilidad SSR — problemas y soluciones

El patrón correcto es usar isPlatformBrowser junto con inject(DOCUMENT) para acceder al documento de forma segura:

```
constructor(@Inject(DOCUMENT) private document: Document) {}

getCsrftoken(): string {
  if (!isPlatformBrowser(this.platformId)) return '';
  return this.document.cookie
    .split(';')
    .find(c => c.trim().startsWith('csrftoken='))
    ?.split('=')[1] ?? '';
}
```

Código 4.8. getCsrftoken — lectura segura con isPlatformBrowser

Para componentes que usan librerías que acceden directamente al DOM, como ApexCharts, la solución fue salir anticipadamente del método de renderizado si el código se está ejecutando en el servidor:

```
renderChart(): void {  
    if (!isPlatformBrowser(this.platformId)) return;  
    // renderizado del gráfico...  
}
```

Código 4.9. renderChart — guard SSR para ApexCharts

Los casos de localStorage sin guard no han causado errores en la práctica porque los componentes que los usan solo se ejecutan tras una interacción del usuario. Son deuda técnica latente que habría que corregir antes de un despliegue en producción.

4.5.- PRUEBAS/IMPLANTACIÓN

4.5.1.- Pruebas funcionales

Se han realizado pruebas manuales de los flujos principales de la aplicación para verificar que cada funcionalidad se comporta según los requisitos definidos. Las pruebas se realizaron tanto en la versión web (navegador Chrome) como en la versión Android.

Los flujos verificados incluyen:

1. Navegación completa por todas las secciones sin sesión iniciada.
2. Registro de nuevo usuario y login posterior.
3. Añadir y eliminar equipos de favoritos.
4. Verificación de la vista de favoritos enriquecida con datos reales.
5. Acceso a la ficha completa de un partido (eventos, estadísticas, alineaciones, H2H).
6. Navegación por el bracket de la Champions League.
7. Visualización de noticias y acceso al detalle.
8. Completar el minijuego del día.
9. Prueba de la aplicación Android en dispositivo físico.
10. Verificación del panel de administración con usuarios de rol admin.

4.5.2.- Pruebas con el modo mock

Una parte importante de las pruebas se realizó con `MOCK_API_MODE = True`, lo que permite verificar el comportamiento de todos los componentes sin depender del estado actual de la API. Los archivos de mock en `livesports/mocks/` cubren los endpoints principales y permiten simular respuestas de distintas ligas y competiciones.

4.5.3.- Implantación local

La aplicación se ejecuta en entorno local con la siguiente configuración:

- Backend Django: `python manage.py runserver 0.0.0.0:8000`
- Frontend Angular: `npm start` (puerto 4200)
- Base de datos MySQL: XAMPP con el servidor MySQL activo en puerto 33306
- App Android: APK generada con Android Studio, conectada al backend por IP local.

Para instalar el proyecto en un nuevo equipo: clonar el repositorio, crear el entorno virtual Python e instalar dependencias con `pip install -r requirements.txt`, instalar dependencias del frontend con `npm install`, crear la base de datos besoccer en MySQL y ejecutar las migraciones con `python manage.py migrate`.

Capítulo 5

Conclusiones y trabajo futuro

5.1.- CONCLUSIONES

5.1.1.- Conclusiones

El objetivo principal de SportsDB era desarrollar una plataforma web, y posteriormente una app Android, que permitiera consultar clasificaciones, resultados, partidos en directo, fichas de equipos y jugadores, y gestionar equipos favoritos, todo centralizado en una interfaz limpia y sin depender de una sola fuente de datos externa. Ese objetivo se ha cumplido. Repasando los objetivos específicos planteados al inicio del proyecto:

- Aprender Angular de forma autodidacta. Cumplido. Cuando empecé este proyecto no había tocado Angular en profundidad. El salto de PHP + JavaScript plano a TypeScript + Angular 17 con componentes standalone, inyección de dependencias,

RxJS y SSR ha sido el aprendizaje técnico más importante de todo el grado para mí. No ha sido fácil —hubo semanas enteras bloqueado con cosas que ahora parecen obvias— pero el resultado lo justifica.

- Implementar una arquitectura cliente-servidor moderna. Cumplido. La separación entre el frontend Angular y el backend Django, siguiendo el patrón BFF, es una arquitectura que se usa en entornos profesionales reales. Entender por qué Angular no debe llamar directamente a las APIs externas, y qué gana el sistema cuando esa lógica reside en el servidor, es algo que no se aprende en un libro.
- Integrar APIs externas de datos deportivos. Cumplido. La integración con LiveScore API y Wikipedia REST API funciona correctamente. El modo mock con `MOCK_API_MODE` ha sido una decisión que ahorra mucho tiempo de desarrollo y que en producción marcaría la diferencia ante caídas o cambios en los proveedores.
- Desarrollar un sistema de gestión de usuarios. Cumplido. El registro, login, gestión de favoritos con datos enriquecidos y el panel de administración de Django funcionan según lo previsto. La decisión de usar sesiones de Django en lugar de JWT ha simplificado el desarrollo, aunque en una versión de producción real habría que revisar esa elección.
- Crear una aplicación Android. Cumplido. La APK generada con Capacitor funciona en dispositivo físico y consume el mismo backend que la versión web. Es la funcionalidad que más satisfacción personal me ha dado, porque con un esfuerzo relativamente pequeño sobre el trabajo ya hecho se consigue algo que antes habría requerido aprender Android Studio desde cero, que aunque es un conocimiento que tengo adquirido, hubiera requerido mucho tiempo.
- Añadir funcionalidades propias más allá de los datos deportivos. Cumplido. El gestor de noticias administrable desde Django Admin, el minijuego de trivia de banderas y el bracket eliminatorio son los tres módulos que diferencian SportsDB de ser simplemente un cliente de LiveScore. No son el núcleo de la aplicación, pero le dan personalidad.

5.1.2.- Valoración personal del proceso

Este proyecto ha tenido dos vidas. La primera fue la versión que entregué en Desarrollo de Servicios Web, con Angular en el frontend y PHP en el backend, sin arquitectura clara y con un alcance bastante más limitado. La segunda es esta: mismo proyecto, reescrito prácticamente desde cero con Django, con una arquitectura pensada, con mucho más

cuidado en las decisiones técnicas y con funcionalidades que en la primera versión ni me planteaba.

Esa evolución entre versiones es en sí misma una de las conclusiones más importantes. No porque el resultado sea perfecto —no lo es— sino porque refleja cómo se entiende un proyecto de verdad cuando llevas meses dentro de él. Las decisiones que tomé al principio sobre la estructura de la API, el modelo de datos o la gestión del estado en Angular las habría tomado de forma muy distinta sabiendo lo que sé ahora.

Hay cosas que me han costado más de lo que esperaba. La gestión del CORS y el CSRF entre Angular y Django consumió días enteros antes de funcionar correctamente. El problema de la tabla usuarios_equipos con clave primaria compuesta y el ORM de Django es el ejemplo más concreto de cómo una decisión de diseño de base de datos puede complicar la implementación de formas que no anticipas. Y la integración del SSR de Angular con el resto del proyecto tuvo sus propios conflictos que no siempre encontraban solución en la documentación oficial.

Dicho esto, no cambiaría la elección de tecnologías. Angular y Django son herramientas que se usan en empresas reales, y haber construido algo funcional con ellas tiene un valor práctico inmediato.

5.1.3.- Limitaciones y trabajo futuro

Hay aspectos del sistema que son mejorables y que no llegaron a completarse en el alcance de este TFG, principalmente por tiempo:

- Seguridad de contraseñas. En el entorno de desarrollo actual, las contraseñas se almacenan sin hash en la base de datos. Es una deuda técnica conocida que en producción sería inaceptable. Django tiene un sistema de hashing de contraseñas integrado (make_password / check_password) que habría que activar.
- CSRF global. La gestión del token CSRF se hace manualmente en cada petición POST o DELETE desde Angular. Lo correcto sería un interceptor HTTP global que lo inyecte de forma automática. Es una tarea pendiente que no afecta a la funcionalidad actual pero que sí afectaría a la mantenibilidad conforme crezca el código.
- Datos mock desactualizados. Los archivos JSON en livesports/mocks/ reflejan el estado de la API en un momento concreto. Si la estructura de respuesta de LiveScore cambia, los mocks dejan de ser fieles. En un proyecto con más tiempo, los mocks se regenerarían automáticamente.

- Tests automatizados. Karma y Jasmine están configurados en el frontend, pero no hay tests escritos más allá del scaffolding inicial. Las pruebas realizadas han sido todas manuales. En un entorno profesional, esto sería el primer punto a corregir.
- Notificaciones push. El icono de campana en la vista de favoritos guarda estado en localStorage pero no hay ninguna integración real con un sistema de push notifications. Es una funcionalidad visible en la UI que está a medias.

5.2.- POSIBLES DESARROLLOS FUTUROS

SportsDB nació con la intención de no cerrarse a un único deporte ni a una única plataforma, y ese principio todavía tiene mucho recorrido. A continuación se describen las ampliaciones que considero más relevantes, ordenadas de mayor a menor prioridad técnica y de impacto para el usuario.

5.2.1.- Seguridad y producción

Antes de cualquier ampliación de funcionalidades, hay una serie de cambios técnicos necesarios si la aplicación se desplegara en un entorno real:

- Activar el hashing de contraseñas de Django en el modelo Usuario.
- Reemplazar `CORS_ALLOW_ALL_ORIGINS = True` por una lista blanca de dominios permitidos. Las cuales serían la Ip que debamos permitir en un servidor integrado.
- Mover las claves de API y configuración sensible a variables de entorno, fuera del settings.py.
- Crear un archivo settings_production.py separado con `DEBUG = False` y configuración de servidor adecuada.
- Implementar el interceptor HTTP global de Angular para el token CSRF.

5.2.2.- Mejoras en la gestión de usuarios

- Hash de contraseñas en el modelo actual, utilizando `make_password` de Django.
- Recuperación de contraseña por correo electrónico.
- Perfil de usuario con nombre, avatar y estadísticas de uso.
- Historial de partidos vistos o seguimiento de competiciones concretas, no solo equipos.

- Notificaciones push reales para los equipos favoritos cuando terminan un partido o empiezan a jugar, integrando Firebase Cloud Messaging o un sistema similar.

5.2.3.- Ampliación de contenidos deportivos

- Más datos de jugadores: Estadísticas detalladas por temporada, valor de mercado, historial de clubes. Actualmente la información de los jugadores viene de Wikipedia, que tiene un nivel de detalle limitado. Integrar una API específica de jugadores como TransferMarkt o FBref cambiaría la experiencia por completo.
- Estadísticas avanzadas de equipo: Gráficas de rendimiento a lo largo de la temporada, evolución en la clasificación jornada a jornada, comparativa entre equipos. Detalle que consigue Sofascore.
- Partidos históricos: Un buscador que permita consultar resultados de temporadas anteriores sin límite de antigüedad.
- Cuotas de apuestas: Datos informativos de casas de apuestas integrados en la ficha del partido, como hacen SofaScore o FlashScore. Solo informativo, no funcionalidad de apuestas.

5.2.4.- Mejoras de interfaz y experiencia de usuario

- Modo oscuro: Una opción de configuración que cambie la paleta de colores a tonos oscuros. Es probablemente la mejora de UX más demandada en aplicaciones de este tipo y técnicamente no es compleja con CSS variables.
- Tutorial de primer acceso: Un recorrido guiado que explique las funciones principales al usuario nuevo. Se podría omitir para usuarios que ya lo hayan completado.
- Selección de idioma: Castellano, valenciano e inglés como mínimo. Angular tiene un sistema de internacionalización (i18n) integrado.
- Búsqueda global: Un campo de búsqueda que permita encontrar equipos, partidos o jugadores directamente desde el header, sin tener que navegar por ligas.
- Compartir resultados: Botón para compartir el resultado de un partido directamente en redes sociales desde la ficha del partido.

5.2.5.- Ampliaciones de la app Android

La versión Android actual es funcional pero es básicamente la versión web dentro de un WebView. Hay mejoras específicas para la plataforma nativa que Capacitor permite implementar:

- Notificaciones push nativas para resultados de equipos favoritos.
- Widget de pantalla de inicio con el próximo partido del equipo favorito.
- Modo offline más robusto con caché local de los últimos datos consultados.
- Biometría para el login en lugar de introducir la contraseña cada vez.

5.2.6.- Módulos nuevos

- Quiniela: Permitir al usuario hacer sus predicciones de resultados antes de cada jornada y ver cuántas acierta al final. Requeriría guardar predicciones en la base de datos y compararlas con los resultados reales de la API.
- Comunidad: Un espacio donde los usuarios puedan comentar partidos o jornadas. Requeriría un modelo de comentarios en la base de datos y moderación.
- Versión premium: Acceso a más ligas, estadísticas avanzadas y funcionalidades extra para usuarios con suscripción. La arquitectura actual ya distingue entre usuarios anónimos e identificados, así que añadir un rol premium sería el siguiente paso natural.
- Inclusión de otros deportes: El nombre SportsDB, y no FootballDB, no es casual. La arquitectura BFF está pensada para que añadir una nueva fuente de datos sea relativamente sencillo. Baloncesto, tenis o deportes de motor serían candidatos naturales.

5.2.7.- Infraestructura y despliegue

Actualmente SportsDB solo funciona en entorno local. Para que fuera accesible públicamente haría falta:

- Despliegue del backend Django en un servidor VPS (DigitalOcean, Hetzner o similar).
- Despliegue del frontend Angular en un CDN o servidor estático.
- Configuración de HTTPS con certificado SSL.
- Base de datos MySQL en servidor dedicado o servicio gestionado.
- Publicación de la APK Android en Google Play Store.

Nada de esto es técnicamente bloqueante, pero requiere tiempo y coste operativo que están fuera del alcance del TFG.



Bibliografía

6.1.- FRAMEWORKS Y LENGUAJES DE PROGRAMACIÓN

- [1] Google LLC, "Angular Documentation — Angular 17"
<https://angular.dev/overview>
01/03/2025

- [2] Google LLC, "Angular — Standalone Components"
<https://angular.dev/guide/components/importing>
01/03/2025

- [3] Google LLC, "Angular — Server-Side Rendering (SSR) with Angular Universal"
<https://angular.dev/guide/ssr>
01/03/2025

- [4] Microsoft Corporation, "TypeScript — JavaScript With Syntax For Types"
<https://www.typescriptlang.org/docs/>
01/02/2025
- [5] ReactiveX, "RxJS — Reactive Extensions Library for JavaScript"
<https://rxjs.dev/guide/overview>
01/02/2025
- [6] Django Software Foundation, "Django 5.2 Documentation"
<https://docs.djangoproject.com/en/5.2/>
01/02/2025
- [7] T. Christie et al., "Django REST Framework — Web APIs for Django"
<https://www.django-rest-framework.org/>
01/02/2025
- [8] Python Software Foundation, "Python 3 Documentation"
<https://docs.python.org/3/>
01/02/2025

6.2.- LIBRERÍAS Y HERRAMIENTAS DE FRONTEND

- [9] The Bootstrap Team, "Bootstrap 5 — The world's most popular CSS framework"
<https://getbootstrap.com/docs/5.3/>
01/02/2025
- [10] Google LLC, "Angular Material — UI component infrastructure and Material Design for Angular"
<https://material.angular.io/>
01/03/2025
- [11] ApexCharts, "ApexCharts.js — Open Source JavaScript Charts",
<https://apexcharts.com/docs/installation/>
01/04/2025

6.3.- PLATAFORMAS MÓVILES

- [12] Ionic Framework Team, "Capacitor — Cross-platform native runtime for web apps"
<https://capacitorjs.com/docs>
01/03/2025
- [13] Google LLC, "Android Developers — Debug your app"
<https://developer.android.com/studio/debug>.
01/04/2025
- [14] Google LLC, "Chrome DevTools — Remote Debugging Android Devices"
<https://developer.chrome.com/docs/devtools/remote-debugging/>
01/04/2025

6.4.- APIs EXTERNAS

- [15] Live Score API Ltd., "LiveScore API — Football Data API Documentation"
<https://live-score-api.com/>
01/02/2025
- [16] Wikimedia Foundation, "MediaWiki Action API"
https://www.mediawiki.org/wiki/API:Main_page
01/03/2025
- [17] Wikimedia Foundation, "Wikipedia REST API"
https://en.wikipedia.org/api/rest_v1/
01/03/2025

6.5.- BASES DE DATOS Y ENTORNOS DE DESARROLLO

- [18] Oracle Corporation, "MySQL 8.0 Reference Manual"
<https://dev.mysql.com/doc/refman/8.0/en/>
01/01/2025
- [19] Apache Friends, "XAMPP — Apache + MariaDB + PHP + Perl"
<https://www.apachefriends.org/es/index.html>
01/03/2025

- [20] PyMySQL Contributors, "PyMySQL — Pure Python MySQL client"
<https://pypi.org/project/PyMySQL/>
01/01/2025
- [21] OpenJS Foundation, "Node.js — Documentation"
<https://nodejs.org/en/docs/>
01/02/2025
- [22] Microsoft Corporation, "Visual Studio Code — Documentation"
<https://code.visualstudio.com/docs>
01/01/2025
- [23] Git SCM, "Git — Documentation"
<https://git-scm.com/doc>
01/02/2025
- [24] Postman Inc., "Postman Learning Center"
<https://learning.postman.com/docs/getting-started/overview/>
01/02/2025

6.6.- ARQUITECTURA Y PATRONES DE DISEÑO

- [25] Building Microservices: Designing Fine-Grained Systems
S. Newman
2nd ed. Sebastopol, CA: O'Reilly Media, 2021.
- [26] S. Fowler, "BFF — Backends For Frontends pattern"
<https://samnewman.io/patterns/architectural/bff/>
01/03/2025
- [27] OWASP Foundation, "Cross-Site Request Forgery (CSRF)"
<https://owasp.org/www-community/attacks/csrf>
01/03/2025
- [28] OWASP Foundation, "Cross-Origin Resource Sharing (CORS)"
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/07-Testing_Cross_Origin_Resource_Sharing
01/03/2025

- [29] Mozilla Developer Network, "HTTP — CORS (Cross-Origin Resource Sharing)"
<https://developer.mozilla.org/es/docs/Web/HTTP/CORS>
01/03/2025

6.7.- INGENIERÍA DEL SOFTWARE Y METODOLOGÍA

- [30] Software Engineering
I. Sommerville
10th ed. Harlow, UK: Pearson Education, 2016
- [31] Software Engineering: A Practitioner's Approach
R. S. Pressman y B. R. Maxim
9th ed. New York, NY: McGraw-Hill Education, 2019
- [32] Object Management Group, "Unified Modeling Language (UML) Specification, Version 2.5.1"
<https://www.omg.org/spec/UML/2.5.1>
01/03/2025

6.8.- PLATAFORMAS DE REFERENCIA Y ANTECEDENTES

- [33] Flashscore, "Flashscore — Resultados en directo"
<https://www.flashscore.es/>
01/01/2025
- [34] SofaScore, "SofaScore — Live Scores, Fixtures & Standings"
<https://www.sofascore.com/>
01/01/2025
- [35] LiveScore Ltd., "LiveScore — Live Football Scores"
<https://www.livescore.com/>
01/01/2025
- [36] BeSoccer, "BeSoccer — Resultados de fútbol en directo"
<https://es.besoccer.com/>
01/01/2025

- [37] 365Scores, "365Scores — Live Scores, News & Stats"
<https://www.365scores.com/>
01/01/2025

6.9.- TECNOLOGÍAS WEB COMPLEMENTARIAS

- [38] Mozilla Developer Network, "HTTP — Hypertext Transfer Protocol"
<https://developer.mozilla.org/es/docs/Web/HTTP>
01/02/2025
- [39] ECMA International, "ECMAScript 2023 Language Specification"
<https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>
01/02/2025
- [40] Mozilla Developer Network, "JSON — JavaScript Object Notation"
https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/JSON
01/02/2025
- [41] World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.1"
<https://www.w3.org/TR/WCAG21/>
01/04/2025

Anexo I

Casos de Uso

El presente anexo presenta las tablas para definición exhaustiva de los casos de uso presentados en el apartado 4.2.4 de la presente memoria.

C.U.-1	Ver partidos del día
Actores	Todos los usuarios
Descripción	El usuario accede a la página principal y consulta los partidos del día.
Dependencias	-
Precondición	-
Secuencia normal	P1 — Abrir la aplicación. P2 — La página principal carga los partidos del día agrupados por país y competición.
Postcondición	Se muestran los partidos con su estado y marcador actualizados.
Excepciones	Si la API no responde, se carga el mock local
Frecuencia	Alta
Importancia	Alta

Tabla AI.1. C.U.-1 Ver partidos del día

C.U.-2	Navegar por fechas en la página de inicio
Actores	Todos los usuarios
Descripción	El usuario navega a un día anterior o siguiente desde el selector de fecha de la página de inicio.
Dependencias	C.U.-1
Precondición	Estar en la página de inicio
Secuencia normal	P1 — Pulsar el botón de día anterior o siguiente. P2 — La aplicación carga los partidos del día seleccionado.
Postcondición	Se muestran los partidos del día seleccionado.
Excepciones	Fechas muy lejanas pueden devolver sin partidos.
Frecuencia	Media
Importancia	Media

Tabla AI.2. RF-28 C.U.-2 Navegar por fechas en la página de inicio

C.U.-3	Ver clasificación de una liga
Actores	Todos los usuarios
Descripción	El usuario selecciona una liga en el sidebar y consulta su tabla de clasificación.
Dependencias	C.U.-24
Precondición	-
Secuencia normal	P1 — Seleccionar una liga en el sidebar. P2 — La aplicación carga la clasificación actualizada.
Postcondición	Se muestra la tabla de clasificación con posición, puntos, partidos jugados y diferencia de goles.
Excepciones	Si la competición tiene fase de grupos, se muestran los grupos genuinos filtrados.
Frecuencia	Alta
Importancia	Alta

Tabla AI.3. C.U.-3 Ver clasificación de una liga

C.U.-4	Consultar clasificación de temporada anterior
Actores	Todos los usuarios
Descripción	Desde la clasificación actual, el usuario cambia la temporada activa para ver clasificaciones históricas.
Dependencias	C.U.-3
Precondición	Estar en la sección de clasificación.
Secuencia normal	P1 — Seleccionar una temporada anterior en el selector. P2 — La aplicación carga la clasificación de esa temporada.
Postcondición	Se muestra la clasificación histórica seleccionada.
Excepciones	Si la API no dispone de datos de esa temporada, se notifica al usuario.
Frecuencia	Baja
Importancia	Media

Tabla AI.4. C.U.-4 Consultar clasificación de temporada anterior

C.U.-5	Ver resultados de una jornada
Actores	Todos los usuarios
Descripción	El usuario consulta los resultados de la jornada actual o de una jornada seleccionada de la liga activa.
Dependencias	C.U.-24
Precondición	Haber seleccionado una liga.
Secuencia normal	P1 — Navegar a la sección de resultados. P2 — La aplicación carga la jornada actual con partidos finalizados, en directo y pendientes.
Postcondición	Se muestran los partidos de la jornada con marcador y estado.
Excepciones	Si la API devuelve datos de temporadas anteriores para esa jornada, el backend los filtra por fecha.
Frecuencia	Alta
Importancia	Alta

Tabla AI.5. C.U.-5 Ver resultados de una jornada

C.U.-6	Navegar entre jornadas
Actores	Todos los usuarios
Descripción	Desde resultados o partidos próximos, el usuario navega a otra jornada usando el selector.
Dependencias	C.U.-5, C.U.-7
Precondición	Estar en la sección de resultados o partidos.
Secuencia normal	P1 — Usar el selector de jornada para elegir otra. P2 — La aplicación carga los partidos de esa jornada.
Postcondición	Se muestran los partidos de la jornada seleccionada.
Excepciones	-
Frecuencia	Media
Importancia	Media

Tabla AI.6. C.U.-6 Navegar entre jornadas

C.U.-7	Ver partidos próximos
Actores	Todos los usuarios
Descripción	El usuario consulta los partidos pendientes de la próxima jornada de la liga activa.
Dependencias	C.U.-24
Precondición	Haber seleccionado una liga.
Secuencia normal	P1 — Navegar a la sección de partidos. P2 — La aplicación carga la siguiente jornada con fecha, hora y equipos de cada partido.
Postcondición	Se muestran los próximos partidos.
Excepciones	Se muestran los próximos partidos.
Frecuencia	Alta
Importancia	Alta

Tabla AI.7. C.U.-7 Ver partidos próximos

C.U.-8	Ver partidos en directo
Actores	Todos los usuarios
Descripción	El usuario consulta los partidos que se están disputando en ese momento con marcador y minuto actual.
Dependencias	C.U.-24
Precondición	Haber seleccionado una liga.
Secuencia normal	P1 — Navegar a la sección de en directo. P2 — La aplicación consulta la API y muestra los partidos activos.
Postcondición	Se muestran los partidos en vivo o se notifica que no hay partidos activos.
Excepciones	Si la API devuelve success: false, se interpreta como sin partidos activos (no como error).
Frecuencia	Alta
Importancia	Alta

Tabla AI.8. C.U.-8 Ver partidos en directo

C.U.-9	Ver goleadores de una liga
Actores	Todos los usuarios
Descripción	El usuario consulta el ranking de máximos goleadores de la liga activa.
Dependencias	C.U.-24
Precondición	Haber seleccionado una liga.
Secuencia normal	P1 — Navegar a la sección de goleadores. P2 — La aplicación carga el ranking con nombre, equipo y goles de cada jugador.
Postcondición	Se muestra el listado de goleadores.
Excepciones	-
Frecuencia	Media
Importancia	Media

Tabla AI.9. C.U.-9 Ver goleadores de una liga

C.U.-10	Ver eventos de un partido
Actores	Todos los usuarios
Descripción	Desde la ficha de un partido, el usuario consulta la línea de tiempo de eventos.
Dependencias	C.U.-5, C.U.-7, C.U.-8
Precondición	Haber pulsado sobre un partido.
Secuencia normal	P1 — Pulsar sobre un partido. P2 — Seleccionar el tab de eventos. P3 — La aplicación muestra goles, tarjetas, cambios y VAR con minuto y jugador.
Postcondición	Se muestra la línea de tiempo de eventos diferenciando local y visitante.
Excepciones	Si la API no devuelve el campo home_away, el backend lo resuelve cruzando con las alineaciones.
Frecuencia	Alta
Importancia	Alta

Tabla AI.10. C.U.-10 Ver eventos de un partido

C.U.-11	Ver estadísticas de un partido
Actores	Todos los usuarios
Descripción	Desde la ficha de un partido, el usuario consulta las estadísticas comparadas entre local y visitante.
Dependencias	C.U.-5, C.U.-7, C.U.-8
Precondición	Haber pulsado sobre un partido.
Secuencia normal	P1 — Pulsar sobre un partido. P2 — Seleccionar el tab de estadísticas. P3 — La aplicación muestra posesión, remates, faltas, córners y tarjetas.
Postcondición	Se muestran las estadísticas en formato comparativo con barras de progreso.
Excepciones	Si la API no devuelve estadísticas, el tab lo indica.
Frecuencia	Alta
Importancia	Alta

Tabla AI.11. C.U.-11 Ver estadísticas de un partido

C.U.-12	Ver alineaciones de un partido
Actores	Todos los usuarios
Descripción	Desde la ficha de un partido, el usuario consulta los once titulares y los suplentes de cada equipo.
Dependencias	C.U.-5, C.U.-7, C.U.-8
Precondición	Haber pulsado sobre un partido.
Secuencia normal	P1 — Pulsar sobre un partido. P2 — Seleccionar el tab de alineaciones. P3 — La aplicación muestra titulares y suplentes de local y visitante.
Postcondición	Se muestran las alineaciones de ambos equipos.
Excepciones	Si la API no devuelve alineaciones, el tab lo indica.
Frecuencia	Alta
Importancia	Alta

Tabla AI.12. C.U.-12 Ver alineaciones de un partido

C.U.-13	Ver historial H2H entre dos equipos
Actores	Todos los usuarios
Descripción	Desde la ficha de un partido, el usuario consulta el historial de enfrentamientos directos entre los dos equipos.
Dependencias	C.U.-5, C.U.-7, C.U.-8
Precondición	Haber pulsado sobre un partido.
Secuencia normal	P1 — Pulsar sobre un partido. P2 — Seleccionar el tab de H2H. P3 — La aplicación muestra los enfrentamientos anteriores con resultado, fecha y competición.
Postcondición	Se muestra el historial de enfrentamientos entre ambos equipos.
Excepciones	Si la API no devuelve datos H2H, el tab lo indica.
Frecuencia	Media
Importancia	Media

Tabla AI.13. C.U.-13 Ver historial H2H entre dos equipos

C.U.-14	Ver bracket eliminatorio
Actores	Todos los usuarios
Descripción	En competiciones de tipo copa, el usuario accede al cuadro eliminatorio visual con los cruces y resultados de cada ronda.
Dependencias	C.U.-24
Precondición	Haber seleccionado una competición de tipo eliminatoria.
Secuencia normal	P1 — Navegar a la sección de eliminatorias. P2 — La aplicación descarga el historial de la competición, filtra las fases eliminatorias y construye el bracket.
Postcondición	Se muestra el cuadro eliminatorio con rondas de izquierda a derecha.
Excepciones	Si la API devuelve rondas con nombres no estándar, pueden no clasificarse correctamente.
Frecuencia	Media
Importancia	Alta

Tabla AI.14. C.U.-14 Ver bracket eliminatorio

C.U.-15	Ver ficha de equipo
Actores	Todos los usuarios
Descripción	El usuario pulsa sobre un equipo y accede a su ficha completa con plantilla, resultados recientes, próximos partidos e información de Wikipedia.
Dependencias	C.U.-3, C.U.-5, C.U.-7
Precondición	Haber pulsado sobre el nombre o escudo de un equipo.
Secuencia normal	P1 — Pulsar sobre un equipo. P2 — La aplicación carga la ficha del equipo combinando datos de LiveScore API y Wikipedia.
Postcondición	Se muestra la ficha con plantilla, resultados, próximos partidos, palmarés e infobox de Wikipedia.
Excepciones	Si Wikipedia no tiene artículo para ese equipo, la sección de información general se omite.
Frecuencia	Alta
Importancia	Alta

Tabla AI.15. C.U.-15 Ver ficha de equipo

C.U.-16	Ver ficha de jugador
Actores	Todos los usuarios
Descripción	Desde la plantilla de un equipo, el usuario pulsa sobre un jugador y ve su ficha de Wikipedia en un modal.
Dependencias	C.U.-15
Precondición	Estar en la sección de plantilla de un equipo.
Secuencia normal	P1 — Estar en la ficha de un equipo. P2 — Pulsar sobre un jugador en la plantilla. P3 — Se abre un modal con el extracto, imagen, datos biográficos y trayectoria del jugador.
Postcondición	El modal muestra la información del jugador.
Excepciones	Si Wikipedia no tiene artículo para ese jugador, el modal lo indica.
Frecuencia	Media
Importancia	Media

Tabla AI.16. C.U.-16 Ver ficha de jugador

C.U.-17	Registrarse
Actores	Usuario anónimo
Descripción	El usuario crea una cuenta en la aplicación introduciendo nickname y contraseña.
Dependencias	-
Precondición	No tener sesión iniciada
Secuencia normal	P1 — Acceder al formulario de registro. P2 — Introducir nickname y contraseña. P3 — Confirmar el registro.
Postcondición	El usuario queda registrado en la base de datos y se redirige al login.
Excepciones	Si el nickname ya existe, se muestra error de clave duplicada (409).
Frecuencia	Baja
Importancia	Alta

Tabla AI.17. C.U.-17 Registrarse

C.U.-18	Iniciar Sesión
Actores	Usuario anónimo
Descripción	El usuario accede con su nickname y contraseña. La sesión se mantiene mediante cookies de sesión de Django
Dependencias	C.U.-17
Precondición	Tener cuenta registrada
Secuencia normal	P1 — Acceder al formulario de login. P2 — Introducir nickname y contraseña. P3 — Confirmar.
Postcondición	La sesión queda activa y el usuario accede a las funcionalidades de usuario identificado.
Excepciones	Credenciales incorrectas devuelven error 401.
Frecuencia	Media
Importancia	Alta

Tabla AI.18. C.U.-18 Iniciar Sesión

C.U.-19	Cerrar sesión
Actores	Usuario identificado
Descripción	El usuario cierra su sesión activa.
Dependencias	C.U.-18
Precondición	Tener sesión iniciada.
Secuencia normal	P1 — Pulsar el botón de cerrar sesión. P2 — El backend elimina la sesión.
Postcondición	La sesión queda cerrada y el usuario vuelve al modo anónimo.
Excepciones	-
Frecuencia	Baja
Importancia	Media

Tabla AI.19. C.U.-19 Cerrar sesión

C.U.-20	Añadir equipo a favoritos
Actores	Usuario identificado
Descripción	El usuario guarda un equipo en su lista de favoritos pulsando la estrella en la ficha del equipo.
Dependencias	C.U.-15, C.U.-18
Precondición	Tener sesión iniciada y estar en la ficha de un equipo.
Secuencia normal	P1 — Pulsar el icono de estrella en la ficha del equipo. P2 — El backend inserta la relación con SQL crudo en la tabla usuarios_equipos. P3 — La estrella cambia de estado.
Postcondición	El equipo queda guardado como favorito del usuario.
Excepciones	Si el equipo ya es favorito, se devuelve error 409 sin duplicar.
Frecuencia	Media
Importancia	Alta

Tabla AI.20. C.U.-20 Añadir equipo a favoritos

C.U.-21	Eliminar equipo de favoritos
Actores	Usuario identificado
Descripción	El usuario elimina un equipo de su lista de favoritos.
Dependencias	C.U.-20
Precondición	El equipo debe estar en la lista de favoritos del usuario.
Secuencia normal	P1 — Pulsar la estrella activa en la ficha del equipo o en la vista de favoritos. P2 — El backend elimina la relación. P3 — El equipo desaparece de la lista.
Postcondición	El equipo queda eliminado de los favoritos del usuario.
Excepciones	-
Frecuencia	Baja
Importancia	Media

Tabla AI.21. C.U.-21 Eliminar equipo de favoritos

C.U.-22	Ver favoritos enriquecidos
Actores	Usuario identificado
Descripción	El usuario accede a su sección de favoritos y ve para cada equipo el último resultado y el próximo partido en tiempo real.
Dependencias	C.U.-20, C.U.-18
Precondición	Tener al menos un equipo en favoritos.
Secuencia normal	P1 — Navegar a la sección de favoritos. P2 — El backend consulta en paralelo los últimos resultados y próximos partidos de cada equipo favorito. P3 — Se muestran las tarjetas enriquecidas.
Postcondición	Cada equipo favorito muestra su último resultado y próximo partido.
Excepciones	-
Frecuencia	Alta
Importancia	Alta

Tabla AI.22. C.U.-22 Ver favoritos enriquecidos

C.U.-23	Ver sección de noticias
Actores	Todos los usuarios
Descripción	El usuario navega a la sección de noticias y consulta los artículos disponibles o accede al detalle de cada uno
Dependencias	-
Precondición	-
Secuencia normal	P1 — Navegar a la sección de noticias. P2 — Se muestra el listado de artículos. P3 (opcional) — Pulsar sobre una noticia para ver su detalle.
Postcondición	Se muestra el listado o el detalle de la noticia seleccionada.
Excepciones	-
Frecuencia	Media
Importancia	Media

Tabla AI.23. C.U.-23 Ver sección de noticias

C.U.-24	Seleccionar liga en el sidebar
Actores	Todos los usuarios
Descripción	El usuario selecciona una liga o competición en el sidebar lateral. La aplicación carga el contenido correspondiente y el header muestra las pestañas adecuadas al tipo de competición.
Dependencias	-
Precondición	-
Secuencia normal	P1 — Pulsar sobre una liga en el sidebar. P2 — La aplicación consulta el tipo de competición (liga, copa o mixta). P3 — Se carga el contenido y el header se adapta.
Postcondición	El contenido de la liga seleccionada se carga en la vista principal.
Excepciones	-
Frecuencia	Alta
Importancia	Alta

Tabla AI.24. C.U.-24 Seleccionar liga en el sidebar

C.U.-25	Jugar el minijuego del día
Actores	Todos los usuarios
Descripción	El usuario accede al minijuego de trivia del día e intenta identificar las banderas presentadas.
Dependencias	-
Precondición	El administrador ha configurado el juego del día.
Secuencia normal	P1 — Navegar a la sección de juego. P2 — La aplicación carga el juego del día. P3 — El usuario responde. P4 — Las respuestas se validan en cliente. P5 — Se muestra el resultado.
Postcondición	Se muestra el resultado del juego sin peticiones adicionales al backend.
Excepciones	-
Frecuencia	Media
Importancia	Media

Tabla AI.25. C.U.-25 Jugar el minijuego del día

C.U.-26	Gestionar noticias desde el panel de administración
Actores	Administrador
Descripción	El administrador crea, edita o elimina artículos de noticias desde Django Admin.
Dependencias	C.U.-18
Precondición	Tener sesión iniciada con rol admin.
Secuencia normal	P1 — Acceder al panel de administración de Django. P2 — Navegar a la sección de noticias. P3 — Crear, editar o eliminar un artículo.
Postcondición	El cambio se refleja inmediatamente en la sección de noticias de la aplicación.
Excepciones	-
Frecuencia	Baja
Importancia	Alta

Tabla AI.26. C.U.-26 Gestionar noticias desde el panel de administración

C.U.-27	Gestionar el minijuego desde el panel de administración
Actores	Administrador
Descripción	El administrador configura el juego del día desde Django Admin, indicando título, banderas y respuestas válidas.
Dependencias	C.U.-18
Precondición	Tener sesión iniciada con rol admin.
Secuencia normal	P1 — Acceder al panel de administración. P2 — Navegar a la sección de juego. P3 — Crear o editar el juego del día con sus parámetros.
Postcondición	El nuevo juego queda disponible para los usuarios.
Excepciones	-
Frecuencia	Baja
Importancia	Alta

Tabla AI.27. C.U.-27 Gestionar el minijuego desde el panel de administración

C.U.-28	Gestionar usuarios desde el panel de administración
Actores	Administrador
Descripción	El administrador consulta, modifica o elimina usuarios desde Django Admin.
Dependencias	C.U.-18
Precondición	Tener sesión iniciada con rol admin.
Secuencia normal	P1 — Acceder al panel de administración. P2 — Navegar a la sección de usuarios. P3 — Consultar, editar o eliminar el usuario deseado.
Postcondición	El usuario queda modificado o eliminado del sistema.
Excepciones	Eliminar un usuario con favoritos elimina también sus relaciones en cascada.
Frecuencia	Baja
Importancia	Alta

Tabla AI.28. C.U.-28 Gestionar usuarios desde el panel de administración

Anexo II

Manual de usuario

Este manual describe el funcionamiento completo de SportsDB, una plataforma web y móvil para el seguimiento de resultados de fútbol. Está dirigido a cualquier tipo de usuario, independientemente de su nivel técnico, y recorre todas las funcionalidades disponibles tanto en la versión web como en la aplicación Android.

SportsDB no requiere instalación en su versión web. Basta con abrir un navegador y acceder a la URL del servidor. La versión Android requiere instalar la APK en el dispositivo.

La aplicación distingue tres tipos de usuario: usuario anónimo (sin sesión, acceso a todas las secciones públicas), usuario identificado (con sesión iniciada, acceso a favoritos, noticias, minijuego y ligas premium) y administrador (acceso adicional al panel de gestión de contenido).

AII.1.- INTERFAZ GENERAL

Al acceder a SportsDB, el usuario encuentra una interfaz dividida en tres zonas permanentes:

Barra superior (header): visible en todo momento, contiene el logotipo de SportsDB a la izquierda y los controles de sesión a la derecha. Cuando no hay sesión iniciada, aparece un único icono de usuario. Cuando hay sesión activa, aparecen cuatro iconos de acceso rápido (noticias, minijuego, favoritos y cerrar sesión) junto al saludo con el nickname del usuario.

Sidebar izquierdo: panel lateral con el listado de ligas disponibles agrupadas bajo el título "Ligas Destacadas". Al pulsar cualquiera de ellas se carga su contenido en la zona central. El sidebar es visible en la versión web de escritorio. En la versión móvil se sustituye por una barra de navegación inferior.

Área de contenido principal: zona central donde se renderiza la información de la liga activa o la sección navegada. Incluye el breadcrumb de navegación bajo la cabecera de liga.

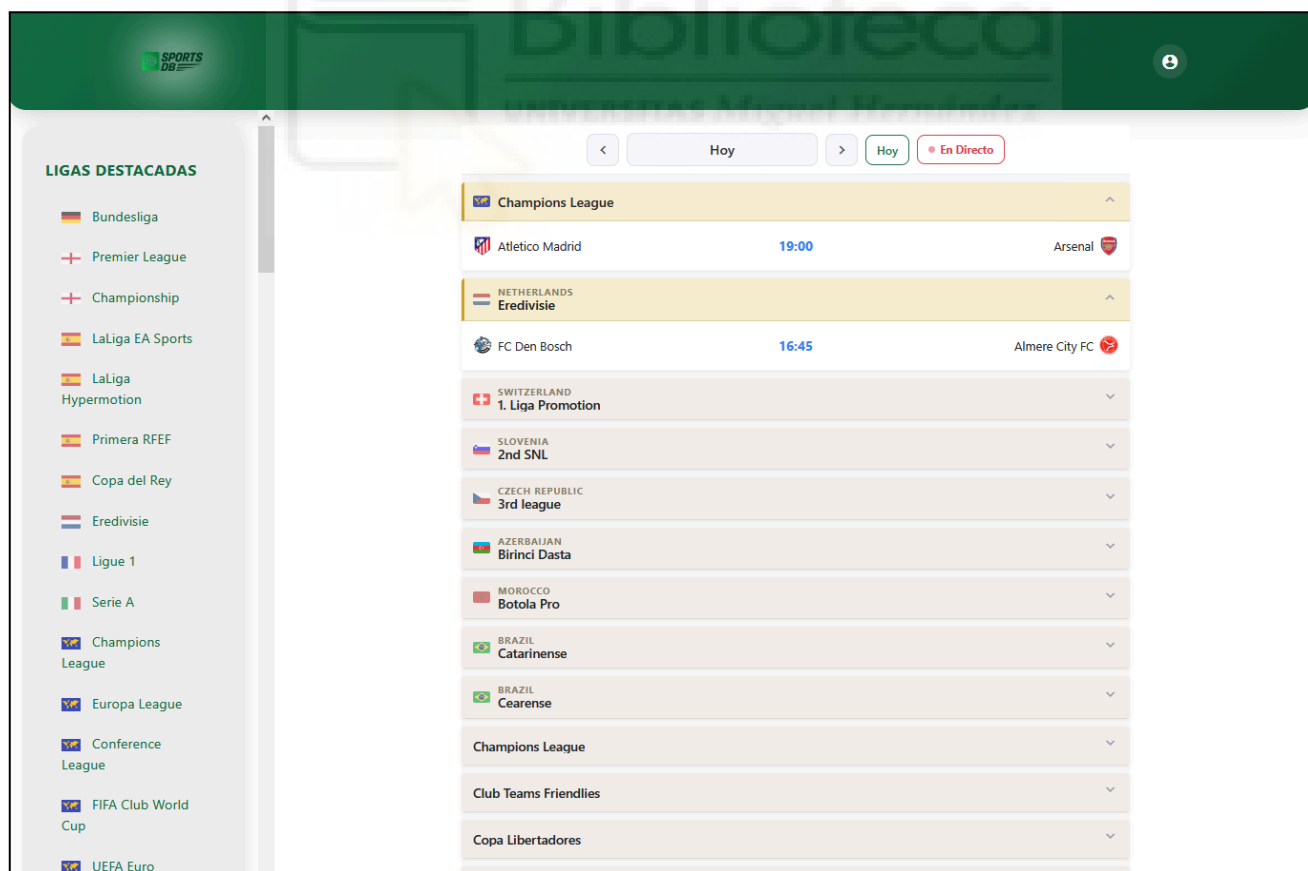


Figura AII.1. Interfaz general de SportsDB

AII.2.- PÁGINA DE INICIO

La página de inicio es la primera pantalla que aparece al abrir la aplicación. Muestra todos los partidos del día agrupados jerárquicamente por país y competición.

Cada grupo de competición muestra la bandera del país, el nombre de la competición y los partidos disponibles. El estado de cada partido se indica visualmente: si el partido ha finalizado, aparece el marcador final; si está en curso, el marcador aparece junto al minuto actual en rojo; si aún no ha comenzado, aparece la hora programada. Tal y como podemos ver en la primera figura del apartado AI.1.

En la parte superior del área de contenido se encuentra el selector de fecha, formado por dos flechas de navegación (día anterior y día siguiente), un indicador con la fecha activa, un botón "Hoy" para volver al día actual, y un botón "En Directo" que filtra la vista para mostrar únicamente los partidos que se están disputando en ese momento.



Figura AII.2 Selector de fecha y botón En Directo

Para navegar a los partidos de otro día, basta con pulsar las flechas del selector. La vista se actualiza automáticamente mostrando los partidos de la fecha seleccionada.

AII.3.- SELECCIONAR UNA LIGA

Desde el sidebar izquierdo, al pulsar sobre cualquier liga, la aplicación carga su contenido en el área central. La cabecera superior cambia para mostrar el nombre de la liga y un menú de pestañas con las secciones disponibles.

Las pestañas varían según el tipo de competición:

- Liga regular (LaLiga, Premier League, Bundesliga...): Clasificación, Resultados, Partidos, En Directo.
- Copa eliminatória (Copa del Rey, FA Cup...): Eliminatorias, En Directo.
- Competición mixta (Champions League, Europa League...): Clasificación, Eliminatorias, En Directo.

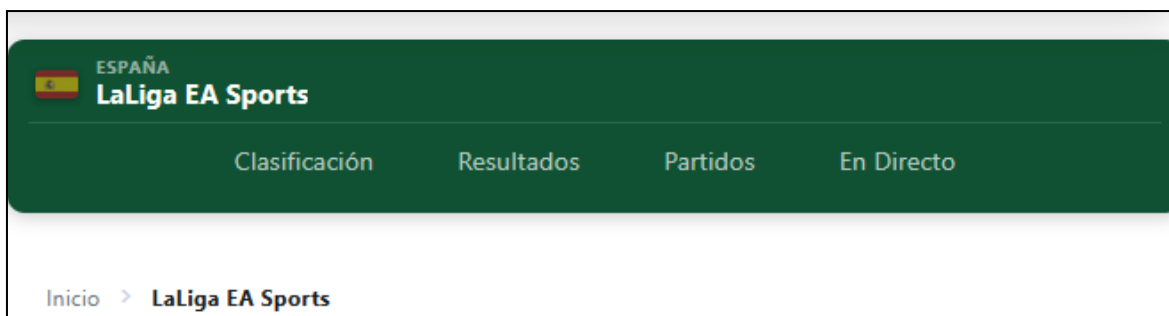


Figura AII.3. Cabecera de liga con pestañas de navegación

El breadcrumb bajo la cabecera indica en todo momento la ruta de navegación activa. Al pulsar sobre cualquier elemento del breadcrumb, el usuario vuelve a ese punto de la navegación.

AII.4.- CLASIFICACIÓN

Al seleccionar una liga y pulsar la pestaña "Clasificación", se muestra la tabla completa de la competición con los siguientes datos por equipo: posición, escudo, nombre, puntos, partidos jugados, victorias, empates, derrotas, goles a favor, goles en contra, diferencia de goles y forma de los últimos cinco partidos (W/D/L con color verde, amarillo y rojo respectivamente).

Pos	#	Equipo	Puntos	PJ	V	E	D	GF	GC	DFG	
1		Barcelona	85	33	28	1	4	87	30	57	W W W W W W
2		Real Madrid	74	33	23	5	5	68	31	37	D W D L W W
3		Villarreal	65	33	20	5	8	59	38	21	W D W L W D
4		Atletico Madrid	60	33	18	6	9	56	37	19	W L L L L W
5		Real Betis	50	33	12	14	7	49	41	8	D W D D L D
6		Getafe	44	33	13	5	15	28	34	-6	L W L W W L
7		Celta Vigo	44	33	11	11	11	45	43	2	L L L W L D
8		Real Sociedad	43	33	11	10	12	52	52	0	D L D W L W
9		Osasuna	42	33	11	9	13	39	40	-1	W L D D W L
10		Athletic Bilbao	41	33	12	5	16	36	48	-12	L W L L W L
11		Rayo Vallecano	39	33	9	12	12	33	41	-8	D W L W L D
12		Valencia	39	33	10	9	14	37	48	-11	W D L L W L
13		Espanyol	39	33	10	9	14	37	49	-12	D L L D L L
14		Elche	38	33	9	11	13	44	50	-6	W W W L W L
15		Girona	38	33	9	11	13	36	50	-14	L L D W L W
16		Alaves	36	33	9	9	15	38	49	-11	W L D D W D

Figura AII.4. Tabla de clasificación con posiciones coloreadas

Las filas se colorean según la posición: los puestos de acceso a Champions League, azul para Europa League, y rojo para los puestos de descenso. Cada liga tiene su propia configuración de colores según el número de plazas europeas y de descenso.

En la esquina superior derecha de la tabla hay un selector de temporada desplegable que permite consultar clasificaciones de temporadas anteriores, todas las que la API ofrece. Al cambiar la temporada, la tabla se recarga automáticamente.

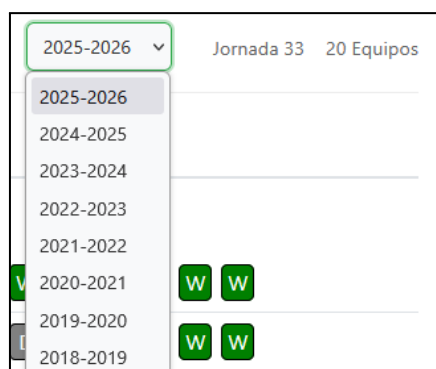


Figura AII.5. Selector de temporada en clasificación histórica

Dentro de la sección de clasificación hay dos pestañas adicionales: Goleadores y Asistentes, accesibles pulsando sobre ellas.

Tabla Goleadores Asistentes					
#	Jugador	Equipo	Goles	Asist.	PJ
1	 Kylian Mbappe	 Real Madrid	24	4	28
2	 Vedat Muriqi	 Mallorca	21	1	32
3	 Lamine Yamal	 Barcelona	16	11	28
4	 Ante Budimir	 Osasuna	16	0	32
5	 Mikel Oyarzabal	 Real Sociedad	14	3	29
6	 Ferran Torres	 Barcelona	14	1	29
7	 Vinicius Junior	 Real Madrid	13	5	32
8	 Robert Lewandowski	 Barcelona	12	2	26
9	 Borja Iglesias	 Celta Vigo	12	2	30
10	 Alexander Sorloth	 Atletico Madrid	12	0	31

Figura AII.6. Pestaña de goleadores

Al pulsar sobre el nombre o escudo de cualquier equipo de la tabla, se accede a su ficha detallada.

AII.5.- RESULTADOS

La sección de Resultados muestra los partidos de la jornada actual ya disputados, así como los que están en curso o aún pendientes de esa misma jornada.

Los partidos se organizan por fecha con separadores visuales. Cada partido muestra el nombre y escudo de local y visitante, y el estado en el centro: marcador final para partidos terminados, minuto en rojo para los que están en directo, y hora para los pendientes.



Jornada 33 ▾		
2026-04-21 · Finalizado		
 Real Madrid	2 - 1	Alaves 
2026-04-21 · Finalizado		
 Girona	2 - 3	Real Betis 
2026-04-21 · Finalizado		
 Mallorca	1 - 1	Valencia 
2026-04-21 · Finalizado		
 Athletic Bilbao	1 - 0	Osasuna 
2026-04-22 · Finalizado		
 Barcelona	1 - 0	Celta Vigo 

Figura AII.7. Sección de resultados por fecha y estado

En la parte superior hay un selector de jornada desplegable que muestra el número de la jornada activa. Al pulsarlo, aparece una lista de todas las jornadas de la temporada. Seleccionando cualquiera de ellas se carga su contenido.



Figura AII.8. Selector de jornada desplegable

Al pulsar sobre cualquier partido de la lista, se accede a su ficha detallada.

AII.6.- PARTIDOS PRÓXIMOS

La sección de Partidos muestra el calendario de la siguiente jornada con los partidos pendientes: fecha, hora y equipos de cada encuentro. Al igual que en Resultados, hay un selector de jornada que permite consultar el calendario de cualquier jornada. Cuando la temporada ha concluido, la aplicación lo indica con un mensaje informativo.

Jornada 34 ▾		
2026-05-01 · 19:00		
 Girona	-	Mallorca 
2026-05-02 · 12:00		
 Villarreal	-	Levante 
2026-05-02 · 14:15		
 Valencia	-	Atletico Madrid 
2026-05-02 · 16:30		
 Alaves	-	Athletic Bilbao 
2026-05-02 · 19:00		
 Osasuna	-	Barcelona 

Figura AII.9. Sección de partidos próximos

AI.7.- EN DIRECTO

La sección "En Directo" muestra únicamente los partidos que se están disputando en ese momento, con el marcador actual y el minuto de juego.



Figura AII.10. Sección de partidos en directo

Si no hay ningún partido en curso en la liga seleccionada, la aplicación muestra un mensaje indicándolo. Al pulsar sobre cualquier partido en directo, se accede a su ficha donde los datos se muestran en tiempo real.

AII.8.- DETALLE DE PARTIDO

Al pulsar sobre cualquier partido (desde Resultados, Partidos o En Directo), se accede a la ficha completa del encuentro. La parte superior muestra una cabecera con el escudo y nombre de ambos equipos, el marcador y el estado (hora, minuto en curso o "FINAL"). Bajo la cabecera, cuatro pestañas dan acceso a los cuatro bloques de información

AII.8.1.- Resumen (eventos)

La pestaña de Resumen muestra la línea de tiempo de todos los eventos del partido: goles, tarjetas amarillas, tarjetas rojas, sustituciones y decisiones VAR. Cada evento indica el minuto en que ocurrió y el jugador implicado. Los eventos del equipo local aparecen a la izquierda y los del visitante a la derecha.

LaLiga Santander · Estadio Santiago Bernabéu	
	
2 - 1	
FINAL	
Estadísticas Alineaciones	
K. Mbappe Assist: A. Guler 30' Aurélien Tchouaméni 35' 1 A. Rudiger 1 Eder Militao 45' Vinicius Junior Assist: F. Valverde 50' 1 F. Mastantuono 1 A. Guler 58' 1 B. Diaz 1 J. Bellingham 58' 1 D. Carvajal 1 T. Alexander-Arnold 63' 1 E. Camavinga 1 A. Tchouameni 63' 1 I. Diabate 1 L. Boye 66' 1 Calebe 1 Yusi 66' 1 A. Guevara 1 A. Blanco 75' Ángel Pérez 90' T. Martinez Assist: A. Guevara 90'	

Figura AII.11. Ficha de partido — tab Resumen (eventos)

Los iconos diferencian el tipo de evento: balón para goles, tarjeta para amonestaciones, flechas para sustituciones y pantalla para el VAR. En las sustituciones se muestran tanto el jugador que entra como el que sale.

AII.8.2.- Estadísticas

La pestaña de Estadísticas muestra las métricas del partido en formato comparativo con barras de progreso: ataques, remates totales, córners, ataques peligrosos, faltas cometidas, tarjetas y otras estadísticas disponibles para ese partido. Los valores del equipo local aparecen a la izquierda y los del visitante a la derecha.

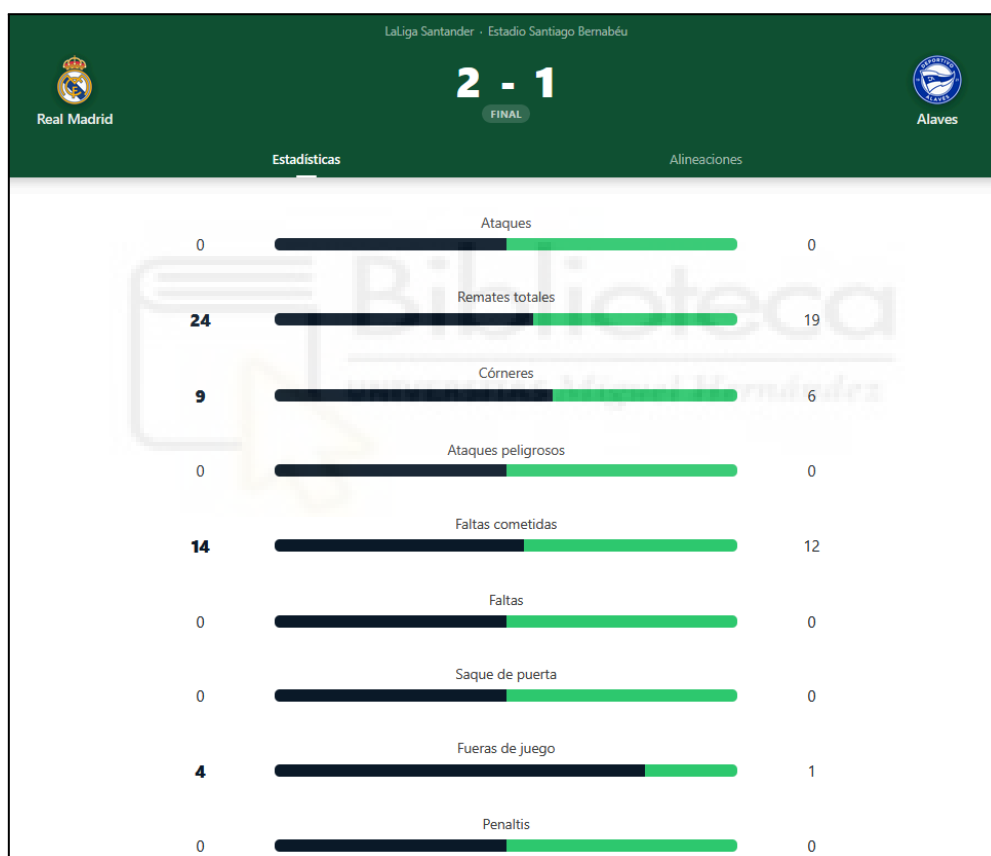


Figura AII.12. Ficha de partido — tab Estadísticas

AII.8.3.- Alineaciones

La pestaña de Alineaciones muestra los once jugadores titulares de cada equipo con su foto, dorsal y nombre, presentados sobre un campo de fútbol visual. Bajo los titulares aparecen los suplentes de cada equipo.

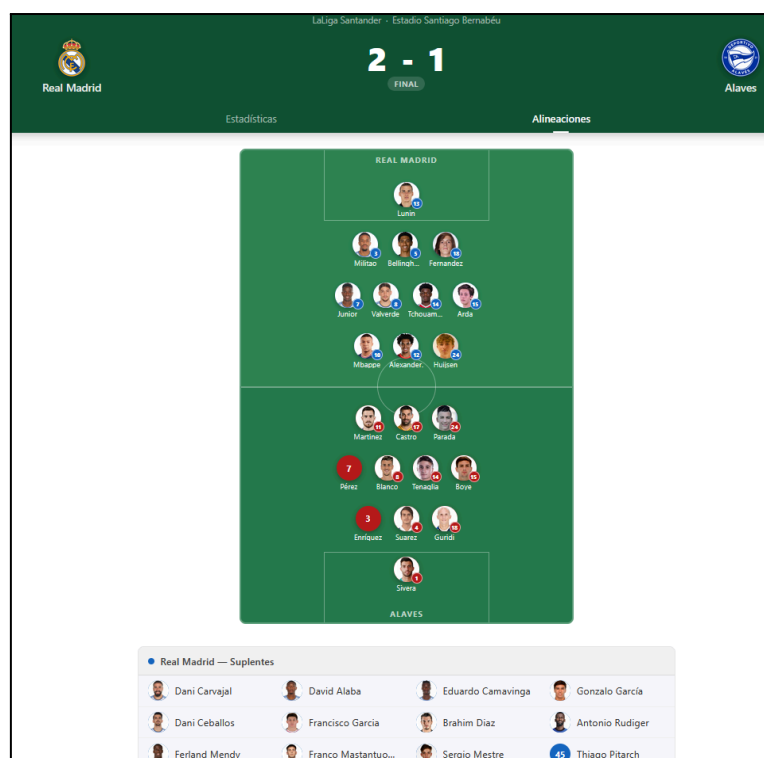


Figura AII.13. Ficha de partido — tab Alineaciones

AII.8.4.- H2H (Head to Head)

La pestaña H2H muestra el historial de enfrentamientos directos entre los dos equipos del partido: resultados anteriores, fechas y competiciones en las que se disputaron.

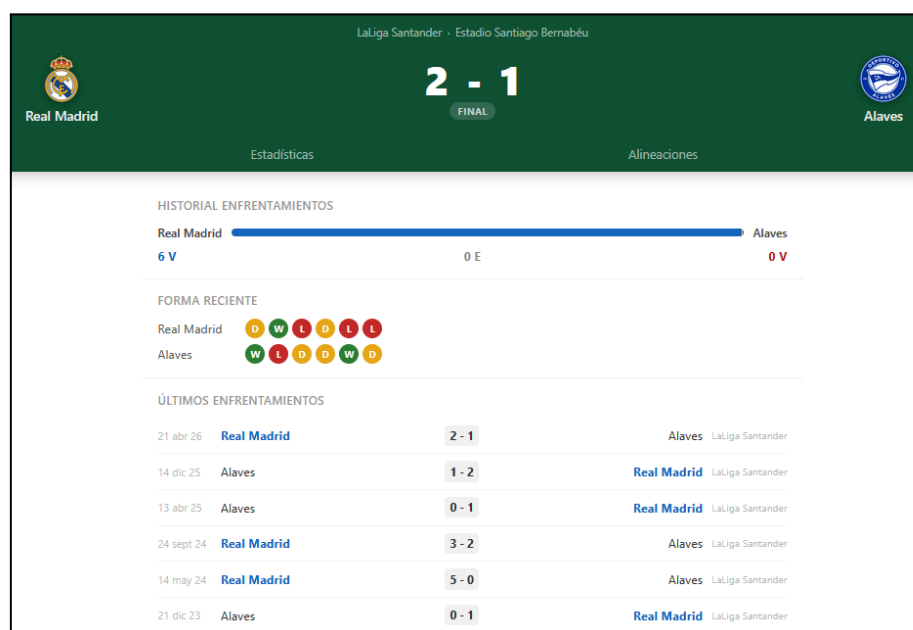


Figura AII.14. Ficha de partido — tab H2H

AII.9.- BRACKET ELIMINATORIO

En competiciones con fase eliminatoria (Champions League, Europa League, Copa del Rey...), la pestaña "Eliminatorias" muestra el cuadro visual del torneo.

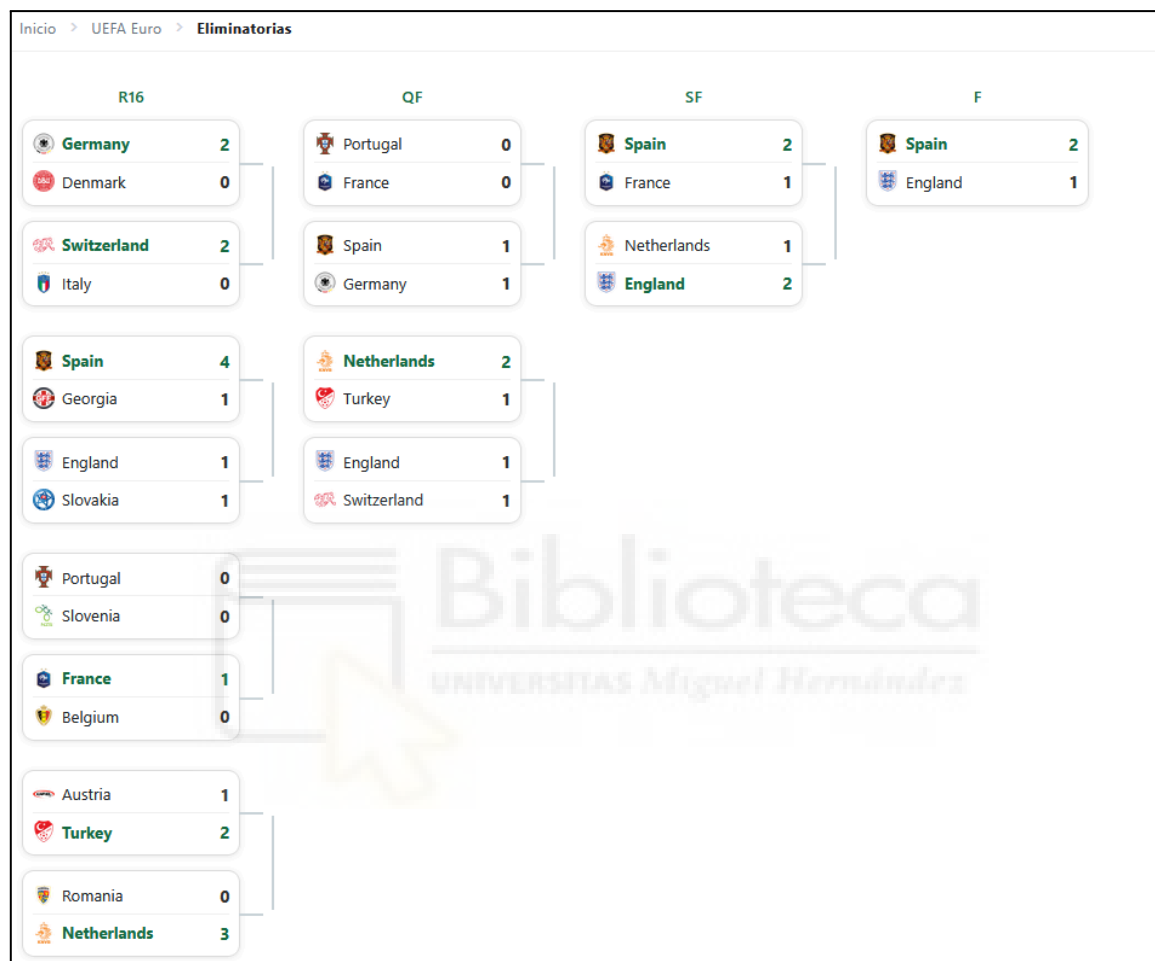


Figura AII.15. Bracket eliminatorio Eurocopa

Las rondas se presentan de izquierda a derecha, desde la primera ronda hasta la final. Cada cruce muestra los dos equipos enfrentados con los resultados de ida y vuelta y el marcador global. Los cruces pendientes de disputarse muestran los equipos sin marcador. Al desplazarse horizontalmente se accede a las rondas más avanzadas.

AII.10.- FICHA DE EQUIPO

Al pulsar sobre el nombre o escudo de cualquier equipo (desde la clasificación, los resultados o el detalle de un partido), se accede a su ficha completa.

La cabecera de la ficha muestra el escudo, el nombre del equipo, el estadio y la forma de los últimos cinco resultados. En la esquina superior derecha aparece una estrella que indica si el equipo está en la lista de favoritos del usuario (dorada si lo está, vacía si no). Solo es visible para usuarios con sesión iniciada. La ficha tiene cuatro pestañas que se indican en los siguientes apartados.

AII.10.1.- Plantilla

Muestra los jugadores del equipo. Cada jugador se presenta con su foto, dorsal y nombre.

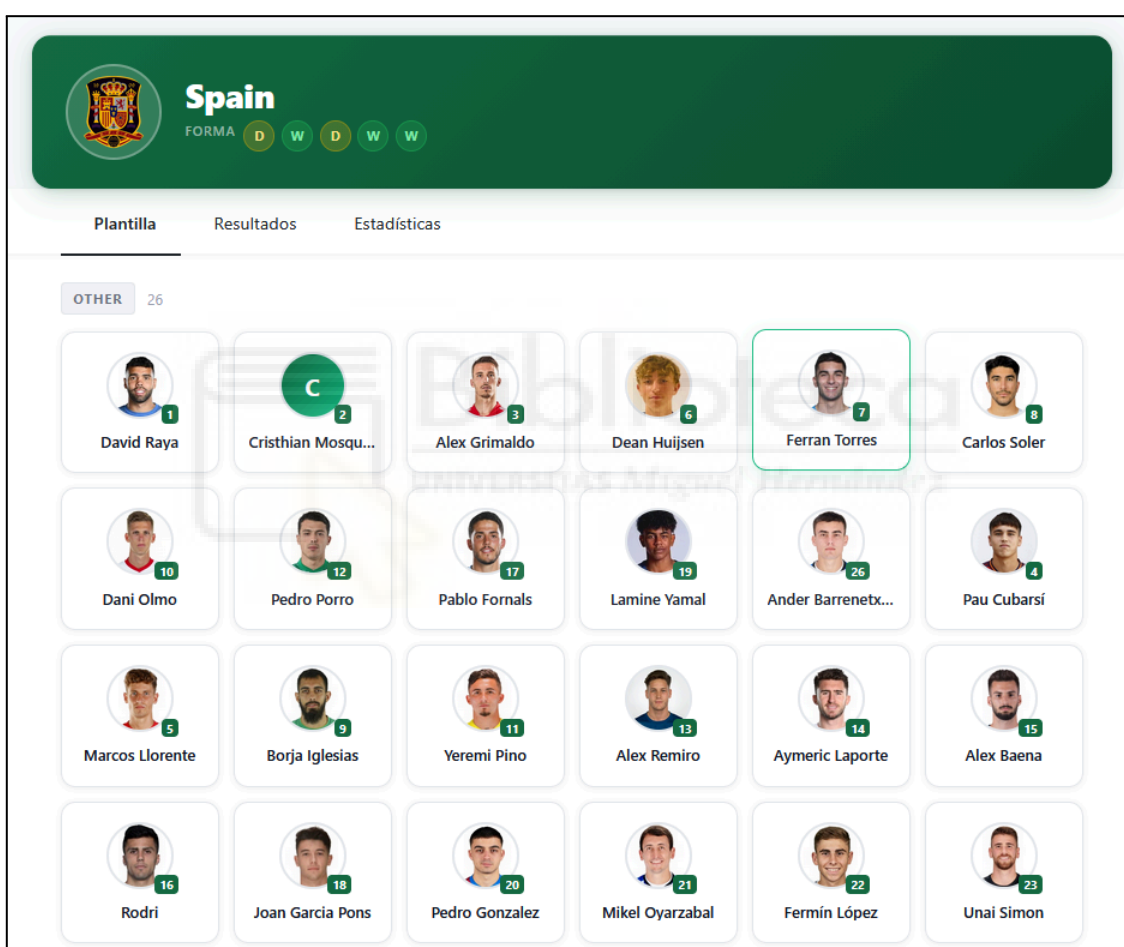


Figura AII.16. Ficha de equipo — tab Plantilla

Al pulsar sobre cualquier jugador se abre un modal con su ficha de Wikipedia: foto, extracto biográfico, datos personales (nacimiento, nacionalidad) y trayectoria profesional. Si Wikipedia no dispone de artículo para ese jugador, el modal lo indica.



Pau Cubarsí

#5

Pau Cubarsí Paredes es un futbolista español que juega como defensa en el F. C. Barcelona de la Primera División de España. Es internacional con la selección de fútbol de España.

DATOS PERSONALES

Nombre completo	Pau Cubarsí Paredes
Nacimiento	Estanyol, Gerona, Cataluña · 22 de enero de 2007 (19 años)
Nacionalidad	Española
Altura	1,83 m
Peso	79 kg

CLUB ACTUAL

Debut	2023 · (F. C. Barcelona Atlètic)
Club	F. C. Barcelona
Posición	Defensa
Dorsal	5
Goles en clubes	2

SELECCIÓN NACIONAL

Selección	España
Debut internacional	22 de marzo de 2024

Figura AII.17. Modal de jugador con información Wikipedia

AII.10.2.- Resultados

Muestra el historial de los últimos partidos del equipo en todas las competiciones, con el resultado, el rival, la fecha y la competición de cada encuentro. Un indicador visual (W/D/L) muestra si fue victoria, empate o derrota desde el punto de vista del equipo.

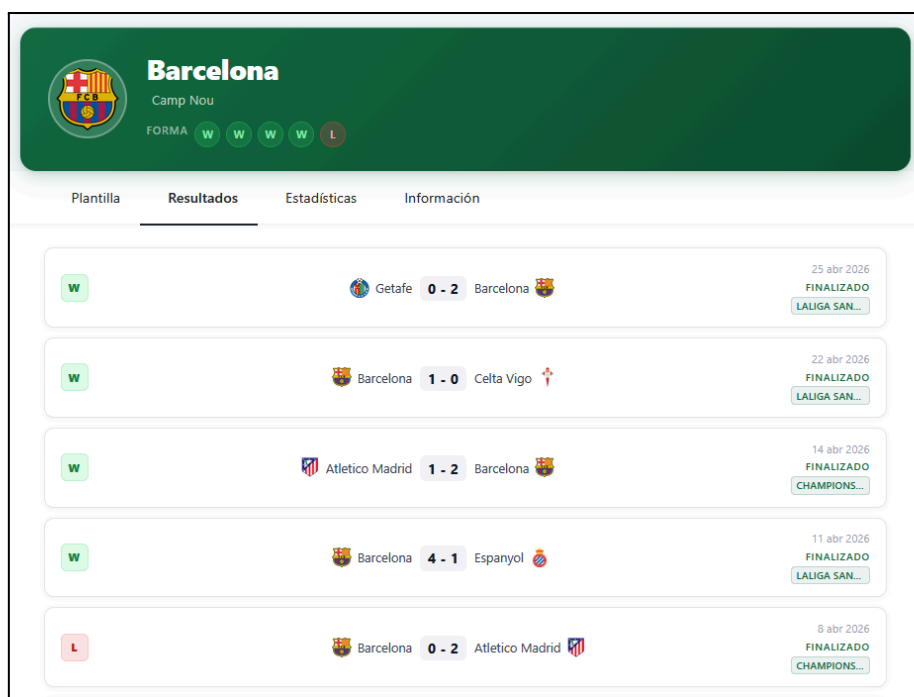


Figura AII.18. Ficha de equipo — tab Resultados

AII.10.3.- Estadísticas

Muestra estadísticas de rendimiento del equipo en la temporada actual mediante gráficos visuales.

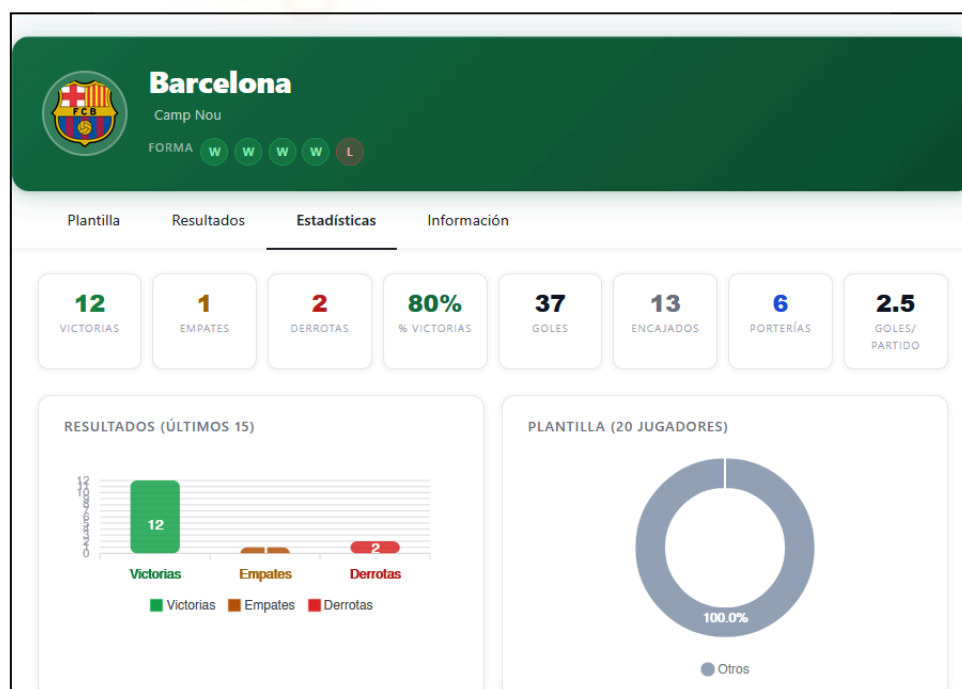


Figura AII.19. Ficha de equipo — tab Estadísticas

AII.10.4.- Información

Muestra los datos del equipo extraídos de Wikipedia: extracto histórico del club con enlace al artículo completo, palmarés desglosado por competición (torneos nacionales e internacionales con número de títulos), y un panel de datos estructurados (infobox) con: nombre completo, fecha de fundación, presidente, entrenador, estadio y capacidad.



Figura AII.20. Ficha de equipo — tab Información Wikipedia

AII.11.- REGISTRO DE USUARIO

Para crear una cuenta en SportsDB, el usuario anónimo puede pulsar el icono de usuario en el header o acceder directamente a cualquier sección que requiera sesión (favoritos, noticias) si está en móvil. La aplicación redirige automáticamente al formulario de login, desde donde hay un enlace al registro.

En el formulario de registro hay que introducir un nickname único y una contraseña. Si el nickname elegido ya existe en el sistema, la aplicación lo notifica con un mensaje de error.

Si el registro es correcto, el usuario queda registrado y se redirige automáticamente al formulario de login.

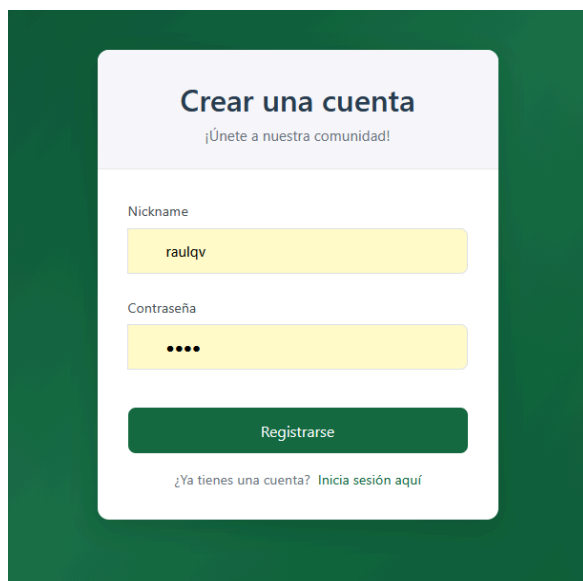
The image shows a registration form titled "Crear una cuenta" (Create an account) with the subtitle "¡Únete a nuestra comunidad!" (Join our community!). It features two input fields: "Nickname" with the text "raulqv" and "Contraseña" (Password) with four dots. Below these is a green "Registrarse" (Register) button. At the bottom, there is a link that says "¿Ya tienes una cuenta? Inicia sesión aquí" (Do you already have an account? Log in here).

Figura AII.21. Formulario de registro

AII.12.- INICIO DE SESIÓN

En el formulario de login, el usuario introduce su nickname y contraseña y pulsa "Iniciar Sesión". Si las credenciales son correctas, la sesión queda activa y el usuario es redirigido a la pantalla desde la que inició el proceso de login.

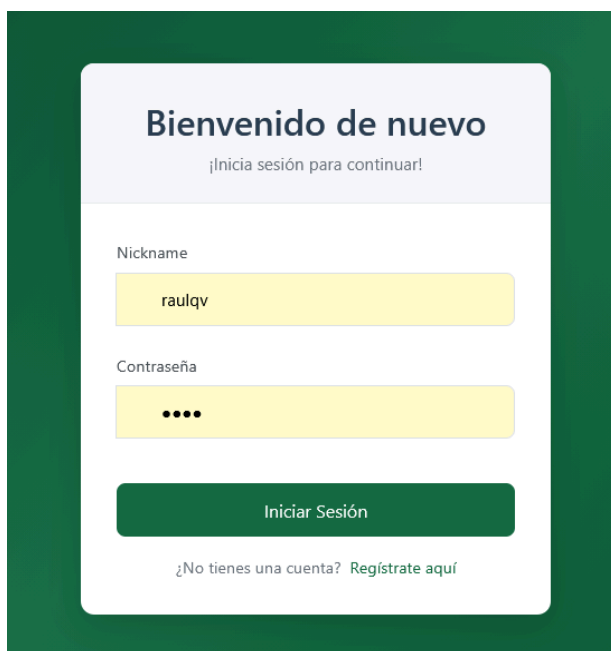
The image shows a login form titled "Bienvenido de nuevo" (Welcome back) with the subtitle "¡Inicia sesión para continuar!" (Log in to continue!). It features two input fields: "Nickname" with the text "raulqv" and "Contraseña" (Password) with four dots. Below these is a green "Iniciar Sesión" (Log In) button. At the bottom, there is a link that says "¿No tienes una cuenta? Regístrate aquí" (Do you not have an account? Register here).

Figura AII.22. Formulario de inicio de sesión

Si las credenciales son incorrectas, la aplicación muestra un mensaje de error. La sesión se mantiene activa mientras no se cierre manualmente, incluso si el usuario cierra y vuelve a abrir el navegador.

Una vez iniciada la sesión, el header cambia: aparecen los cuatro iconos de acceso rápido (noticias, minijuego, favoritos y cerrar sesión) junto al saludo con el nickname del usuario.



Figura AII.23. Header con sesión iniciada

AII.13.- CERRAR SESIÓN

Para cerrar la sesión, el usuario pulsa el icono de salida en el header (último de los cuatro iconos). La sesión se cierra inmediatamente y el usuario vuelve al modo anónimo.

AII.14.- EQUIPOS FAVORITOS

AII.14.1.- Añadir equipo a favoritos

Con sesión iniciada, al acceder a la ficha de cualquier equipo, aparece una estrella en la esquina superior derecha de la cabecera. Si está vacía, el equipo no está en favoritos. Al pulsarla, el equipo se añade a la lista y la estrella pasa a dorado.

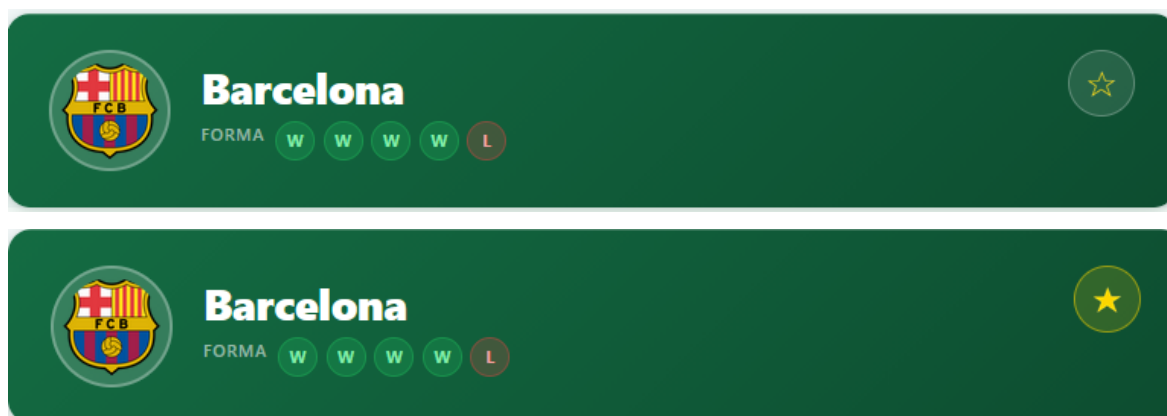


Figura AII.24. Estrella de favorito — inactivo y activo

AII.14.2.- Eliminar un equipo de favoritos

Para eliminar un equipo de favoritos hay dos formas: pulsar la estrella dorada en la ficha del equipo (vuelve a estado vacío), o usar el botón de eliminar en la sección de favoritos.

AII.14.3.- Ver la lista de favoritos

La sección de favoritos es accesible desde el icono de estrella en el header (solo para usuarios con sesión). Muestra todos los equipos guardados como tarjetas expandibles.

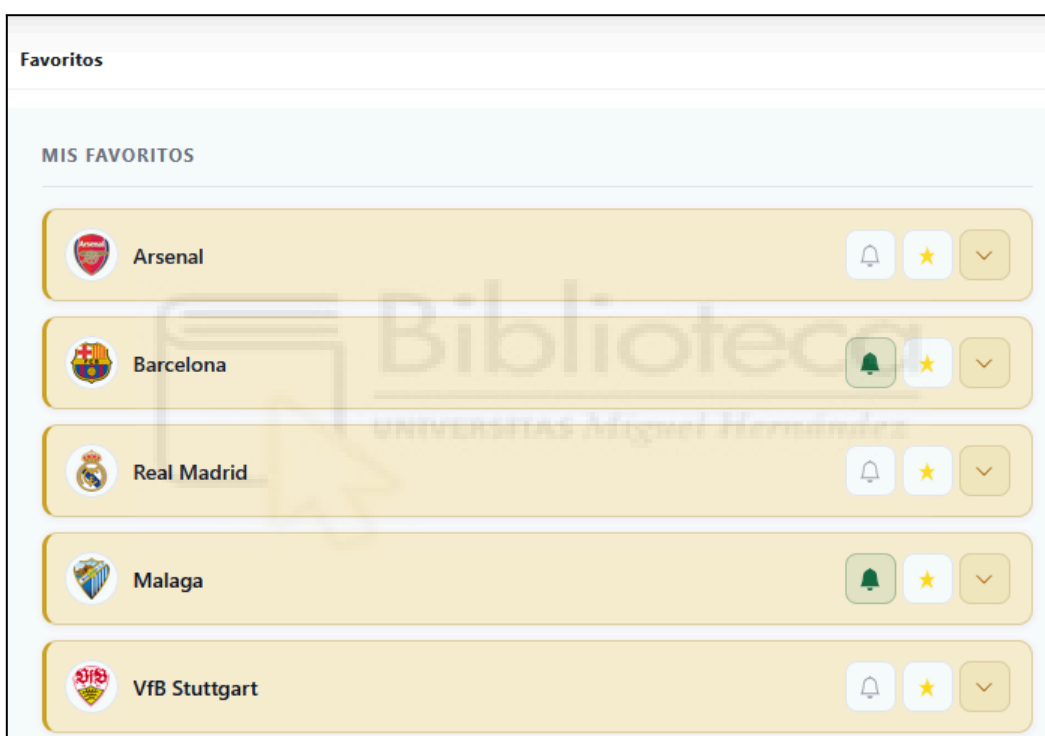


Figura AII.25. Sección de favoritos

Cada tarjeta muestra el escudo y nombre del equipo con la estrella activa. Al pulsar la flecha de expansión, la tarjeta se despliega mostrando dos bloques en paralelo con información en tiempo real:

- Último resultado: el marcador del partido más reciente del equipo, con rival y resultado.
- Próximo partido: la fecha y rival del siguiente encuentro programado.

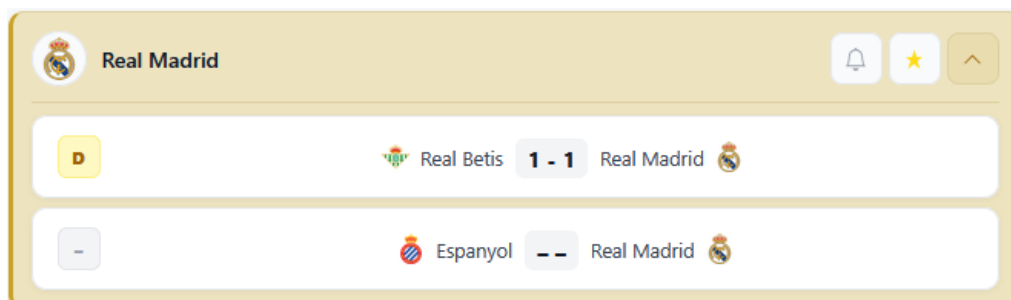


Figura AII.26. Tarjeta de favorito expandida

AII.15.- SECCIÓN DE NOTICIAS

La sección de noticias es accesible desde el icono de noticias en el header para usuarios con sesión iniciada. Muestra un listado de artículos deportivos con imagen, título, autor y fecha.



Figura AII.27. Sección de noticias

Al pulsar sobre cualquier artículo, se accede a su vista de detalle con el contenido completo.

Jorrel Hato insinúa que podría quedarse en el Ajax

Alex Roberts - Tuesday, June 17, 2025



Jorrel Hato (19), objetivo de muchos grandes equipos europeos, ha insistido en que podría quedarse en el Ajax, el club de su infancia, en lugar de buscar un traspaso a la una de las grandes ligas este verano. El futbolista ha sido muy relacionado con la posibilidad de abandonar el club neerlandés este verano, con tres grandes ingleses como el Chelsea, el Arsenal y el Liverpool como interesados. Sin embargo, Hato no está presionando para salir del Ajax, club en el que "se siente como en casa" y con el que debutó a la edad de 16 años en 2023. Sus palabras Hato declaró al sitio web del Ajax: "Lo más inteligente es

Figura AII.28. Vista de detalle de noticia

AII.16.- MINIJUEGO DEL DÍA

El minijuego es accesible desde el icono de mando en el header para usuarios con sesión iniciada. Cada día el administrador configura un desafío diferente.

The image shows a web interface for a daily quiz. At the top, it says 'Inicio > Juego'. The main title is 'TOP 10 JUGADORES DE LA CHAMPIONS LEAGUE SEGÚN SOFASCORE'. Below the title is a progress bar showing '0 / 10'. There is a search bar with the placeholder 'Escribe una respuesta...' and a 'Comprobar' button. Below this are ten numbered input fields for answers, arranged in two columns of five. The numbers 1 through 10 are in green circles. There are also icons for a question mark and a lightbulb.

Figura AII.29. Minijuego del día — estado inicial

La interfaz del juego muestra:

- Título del desafío: describe el reto del día (por ejemplo, "Top 10 jugadores de la Champions League según Sofascore").
- Barra de progreso: indica cuántas respuestas se han acertado sobre el total.
- Campo de texto: donde el usuario escribe su respuesta y pulsa "Comprobar".
- Icono de pista: ayuda opcional que revela información adicional.
- Icono de bandera: permite rendirse y ver todas las respuestas correctas.
- Grid de huecos: muestra las respuestas correctas en verde según se van acertando, y los huecos pendientes en gris.



Figura AII.30. Minijuego en progreso — respuestas acertadas

Las respuestas se validan en el cliente sin necesidad de peticiones adicionales al servidor. El juego termina cuando el usuario acierta todos los huecos o pulsa la bandera para rendirse.

AII.17.- Panel de administración

El panel de administración está disponible exclusivamente para usuarios con rol de administrador. Se accede navegando a la URL /admin/ del servidor backend. Las credenciales de acceso son las mismas que para el login de la aplicación.

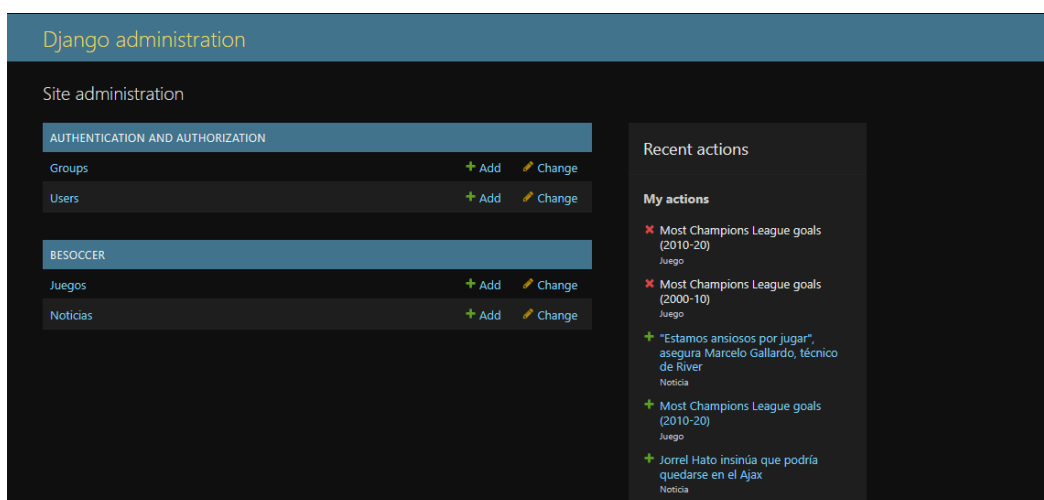


Figura AII.31. Panel de administración Django

Desde el panel de administración se pueden gestionar tres tipos de contenido:

AII.17.1.- Gestión de noticias

En la sección "Noticias", el administrador puede ver el listado de artículos existentes, crear uno nuevo, editar cualquier artículo existente o eliminarlo.

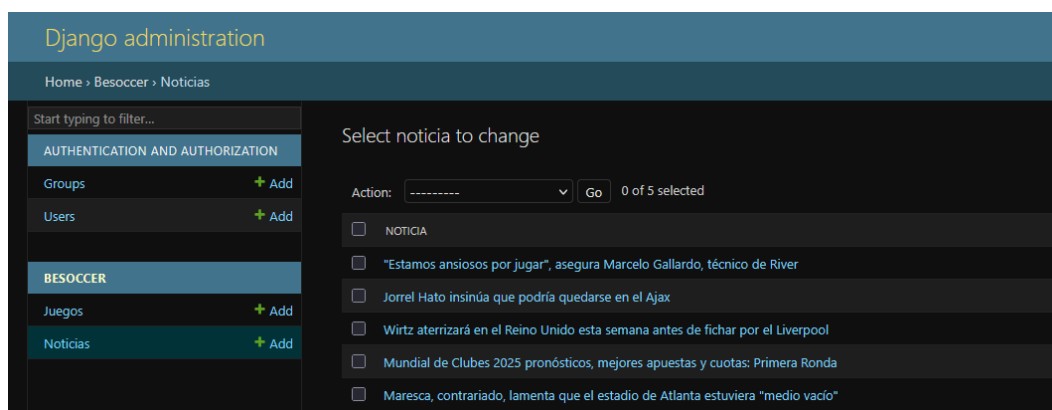


Figura AII.32. Gestión de noticias desde el panel

Para crear una noticia, hay que rellenar los campos: título, imagen (subida desde el equipo local), autor y contenido. Al guardar, la noticia queda publicada inmediatamente en la sección de noticias de la aplicación.

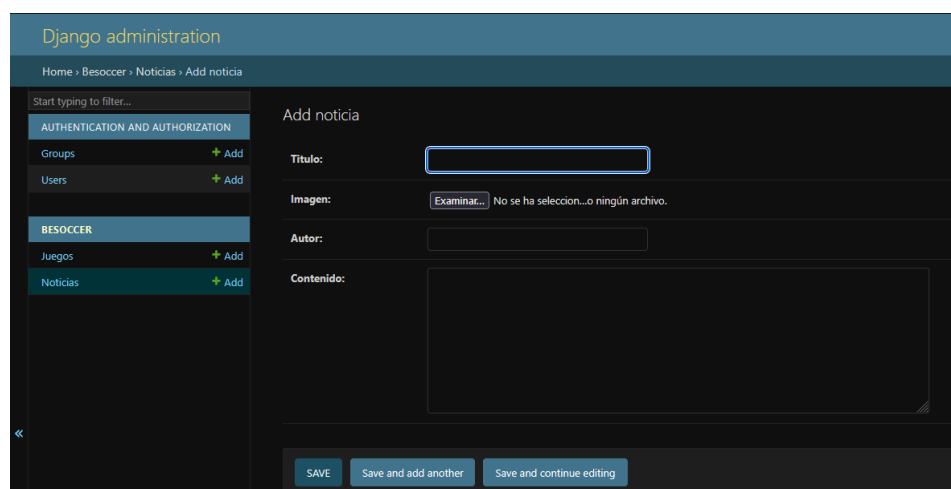
The screenshot shows the Django administration interface. The breadcrumb trail at the top reads 'Home > Besoccer > Noticias > Add noticia'. On the left sidebar, under the 'BESOCCKER' section, 'Noticias' is highlighted with a '+ Add' link. The main content area is titled 'Add noticia' and contains four form fields: 'Titulo:' with a text input, 'Imagen:' with a file upload button labeled 'Examinar...' and a message 'No se ha seleccion...o ningún archivo.', 'Autor:' with a text input, and 'Contenido:' with a large text area. At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Figura AII.33. Añadir noticias desde el panel

AII.17.2.- Gestión de minijuego

En la sección "Juegos", el administrador configura el desafío del día: título descriptivo del reto, las preguntas o pistas en formato JSON, y las respuestas válidas también en formato JSON.

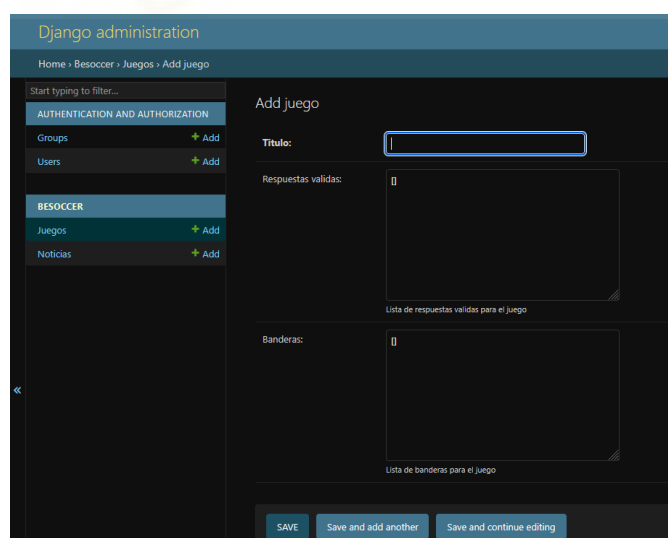
The screenshot shows the Django administration interface for adding a mini-game. The breadcrumb trail at the top reads 'Home > Besoccer > Juegos > Add juego'. On the left sidebar, under the 'BESOCCKER' section, 'Juegos' is highlighted with a '+ Add' link. The main content area is titled 'Add juego' and contains three form fields: 'Titulo:' with a text input, 'Respuestas validas:' with a JSON editor showing an empty array '[]', and 'Banderas:' with a JSON editor showing an empty array '[]'. Below each JSON editor is a small text label: 'Lista de respuestas validas para el juego' and 'Lista de banderas para el juego' respectively. At the bottom of the form are three buttons: 'SAVE', 'Save and add another', and 'Save and continue editing'.

Figura AII.34. Configuración del minijuego desde el panel

Al guardar, el nuevo juego queda disponible para todos los usuarios en la sección de minijuego.

AII.17.3.- Gestión de usuarios

En la sección "Usuarios", el administrador puede consultar el listado completo de cuentas registradas, ver sus datos, modificarlos o eliminar cualquier cuenta.

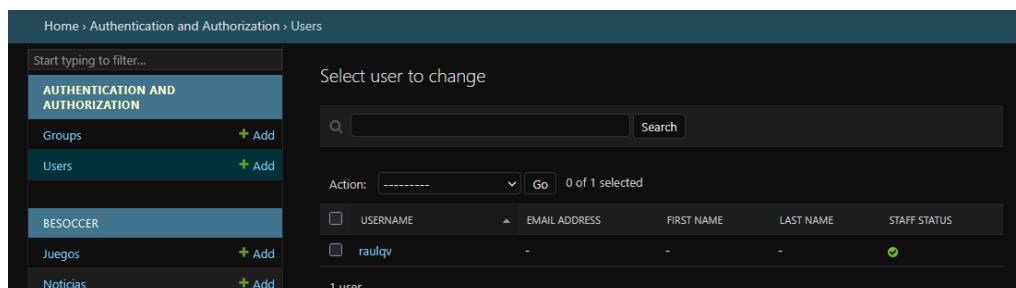


Figura AII.35. Gestión de usuarios desde el panel

AII.18.- APLICACIÓN ANDROID

La versión Android de SportsDB se instala como APK en el dispositivo. Para que funcione, el dispositivo debe estar en la misma red local que el servidor donde corre el backend Django.



Figura AII.36. Pantalla de inicio en dispositivo Android

La interfaz en Android es equivalente a la versión web con los siguientes ajustes:

- El sidebar lateral se sustituye por una barra de navegación inferior con cuatro opciones: Inicio, Ligas, Noticias y Favoritos.
- El header se simplifica: las pestañas de liga se muestran como una fila compacta bajo la barra superior.
- Las tablas de clasificación reducen las columnas visibles para adaptarse a la pantalla pequeña, mostrando solo posición, equipo, puntos, partidos jugados y diferencia de goles.

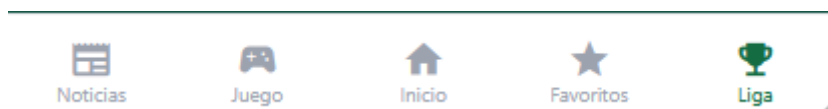


Figura AII.37. Navegación inferior en la versión Android

Todas las funcionalidades descritas en este manual están disponibles en la versión Android: inicio con partidos del día, clasificaciones, resultados, partidos, en directo, bracket, detalle de partido, ficha de equipo, favoritos, noticias y minijuego.

