MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

PlayMatch: diseño e implementación de una aplicación web para la conexión entre profesores y alumnos de pádel

Autor: Amaia Rotaeche Montero

Director: Atilano Ramiro Fernández-Pacheco Sánchez-Migallón

Madrid

junio de 2026

Declaración de originalidad

Declaro bajo mi responsabilidad que el Proyecto presentado con el título **PlayMatch: diseño e implementación de una aplicación web para la conexión entre profesores y alumnos de pádel** en la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico **2025/26** es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial[1]

Declaro bajo mi responsabilidad que (indicar la opción correcta):

☐ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

X He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): "*La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente*". En concreto, las Inteligencia Artificial ha sido empleada para:

He utilizado Inteligencia Artificial en la elaboración del presente documento siempre dentro de las condiciones permitidas por la Universidad Pontificia Comillas, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024). En concreto, la IA ha sido empleada para las siguientes actividades previas o de apoyo a la tarea: • **Brainstorming y estructuración inicial:** explorar posibles enfoques del proyecto, esbozar la organización del repositorio de trabajo y de la memoria y generar ideas sobre funcionalidades del sistema antes de desarrollarlas de forma independiente. • **Identificación de referencias preliminares:** localizar fuentes relevantes sobre tecnologías, marcos regulatorios y trabajos relacionados, usada conjuntamente con otras herramientas y contrastando y validando manualmente los resultados. • **Síntesis de conceptos técnicos:** comprender y resumir protocolos y herramientas específicas —como OAuth2, JWT o Docker Compose— antes de redactar su descripción en el documento. • **Corrección puntual de errores de código:** La asistencia integrada de Visual Studio, "Copilot", para la resolución de errores puntuales, especialmente

[1] Esta declaración se refiere al uso de la Inteligencia Artificial generativa para realizar los documentos del Proyecto (Anexo B y Memoria). No aplica a Proyectos donde, por su naturaleza, deban emplear inteligencia artificial como parte de los mismos (aplicación de técnicas de aprendizaje automático, redes neuronales, análisis de datos…)

relacionados con la lógica de "ubicación" en el algoritmo de matching y el sistema de pagos con retención.

- **Revisión y mejora del estilo:** pulir la redacción de determinados párrafos, manteniendo en todo momento el contenido técnico y las decisiones de diseño como responsabilidad exclusiva del autor.
- **Traducción y adaptación de textos:** trasladar fragmentos del inglés al español en el proceso de documentación.

Firmado (alumno): Amaia Rotaeche Montero

Fecha: 09/06/2026

Autorización para la entrega del Proyecto

El Director del Proyecto	El co-Director del Proyecto (si aplica)
Atilano Ramiro Fernández-Pacheco Sánchez-Migallón 7121631 4B ATILANO Signed by: 71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO	**NO aplica**
Fdo:	Fdo:
Fecha:	Fecha:

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

PlayMatch: diseño e implementación de una aplicación web para la conexión entre profesores y alumnos de pádel

Autor: Amaia Rotaeche Montero

Director: Atilano Ramiro Fernández-Pacheco Sánchez-Migallón

Madrid

PLAYMATCH: DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA LA CONEXIÓN ENTRE PROFESORES Y ALUMNOS DE PÁDEL

Autor: Rotaeche Montero, Amaia.
Director: Férnadez-Pacheco Sánchez-Migallón, Atilano Ramiro
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

PlayMatch es una plataforma web de reserva de clases de pádel desarrollada como TFM. El backend está construido en Python/FastAPI con una base de datos relacional, el frontend en React, y el despliegue en producción usa Docker Compose. La memoria documenta el ciclo completo: análisis de requisitos, diseño UML, implementación de la API REST, pruebas con pytest (78% de cobertura) y planificación del proyecto.

Palabras clave: pádel, aplicación web, algoritmo de matching, FastAPI, React, escrow, producto mínimo viable.

1. Introducción

El pádel se ha consolidado como uno de los deportes de mayor crecimiento en España, superando los 35 millones de jugadores a escala mundial en 2025 (FIP, 2025) y con más de 109.000 licencias federativas en España (Consejo Superior de Deportes, 2024). Este crecimiento ha multiplicado la demanda de clases particulares impartidas por entrenadores que trabajan de forma autónoma. Sin embargo, la relación entre alumno y entrenador sigue gestionándose a través de canales informales y herramientas dispersas, lo que genera fricción operativa, dificulta el seguimiento de la actividad e introduce desconfianza en la transacción económica.

2. Definición del proyecto

El presente Trabajo Fin de Máster aborda este problema mediante el diseño y la implementación de PlayMatch, una aplicación web full-stack que estructura y digitaliza dicha relación, centralizando la oferta y la demanda y dando soporte al ciclo completo de la clase particular: búsqueda y matching, reserva, pago en retención, comunicación y valoración.

3. Descripción del sistema

PlayMatch se ha construido como una aplicación full-stack con un backend en FastAPI sobre una base de datos relacional gestionada con SQLAlchemy y Alembic, y un frontend de página única (SPA) en React y TypeScript. La arquitectura sigue un modelo en capas (presentación, servicios, persistencia) con separación de responsabilidades, API REST documentada con OpenAPI y contenedorización mediante Docker Compose para el despliegue reproducible.

El núcleo técnico del sistema es un algoritmo de matching que recomienda a cada alumno las clases disponibles que mejor encajan con su perfil. El algoritmo opera en dos etapas: (1) filtrado por ciudad, disponibilidad y vigencia, y (2) una función de puntuación que combina la distancia geográfica calculada con la fórmula de Haversine (Sinnott, 1984), la diferencia de nivel y la penalización por precio, produciendo una recomendación transparente, explicable y reproducible. En torno al matching se articulan la gestión de la disponibilidad, un ciclo de reservas con pago en retención (escrow), el sistema de comunicación, las valoraciones multidimensionales, la gestión de incidencias y un entrenador virtual determinista.

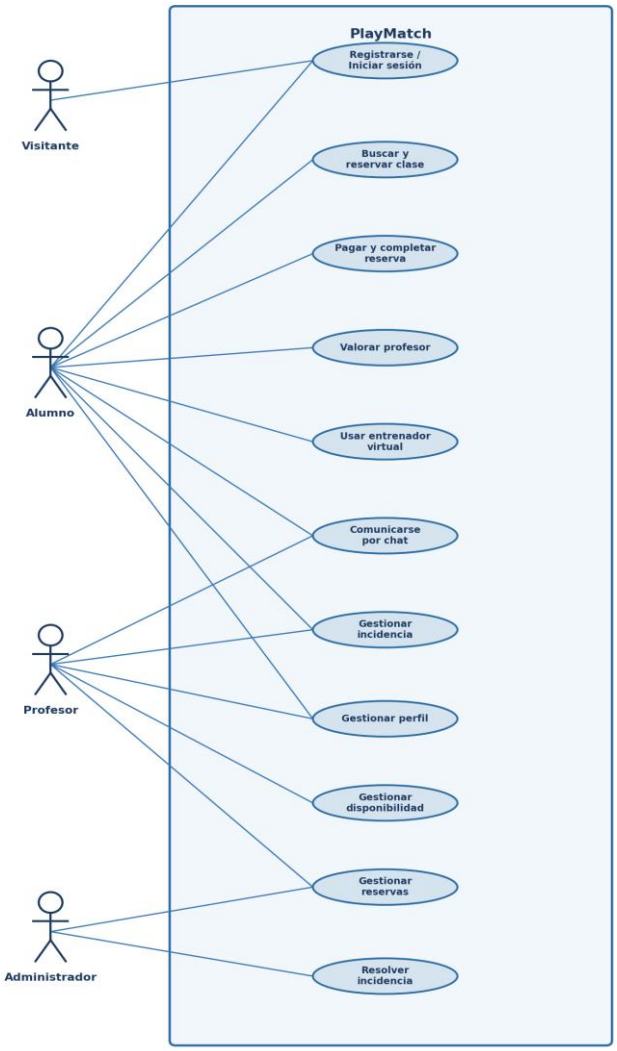

Ilustración 1: Diagrama de casos de uso de PlayMatch

4. Resultados

El resultado del trabajo es un prototipo funcional y validado que integra todos los módulos planificados: autenticación y control de acceso por rol, gestión de perfiles, publicación de disponibilidad, matching y búsqueda, reservas con escrow, chat, valoraciones, incidencias y entrenador virtual. La validación se ha realizado mediante pruebas automatizadas (pytest) con una cobertura global del 78 %, con los módulos críticos (auth: 91 %, matching: 88 %) por encima de la media. La validación funcional ha verificado los flujos principales de los tres roles (alumno, profesor, administrador).

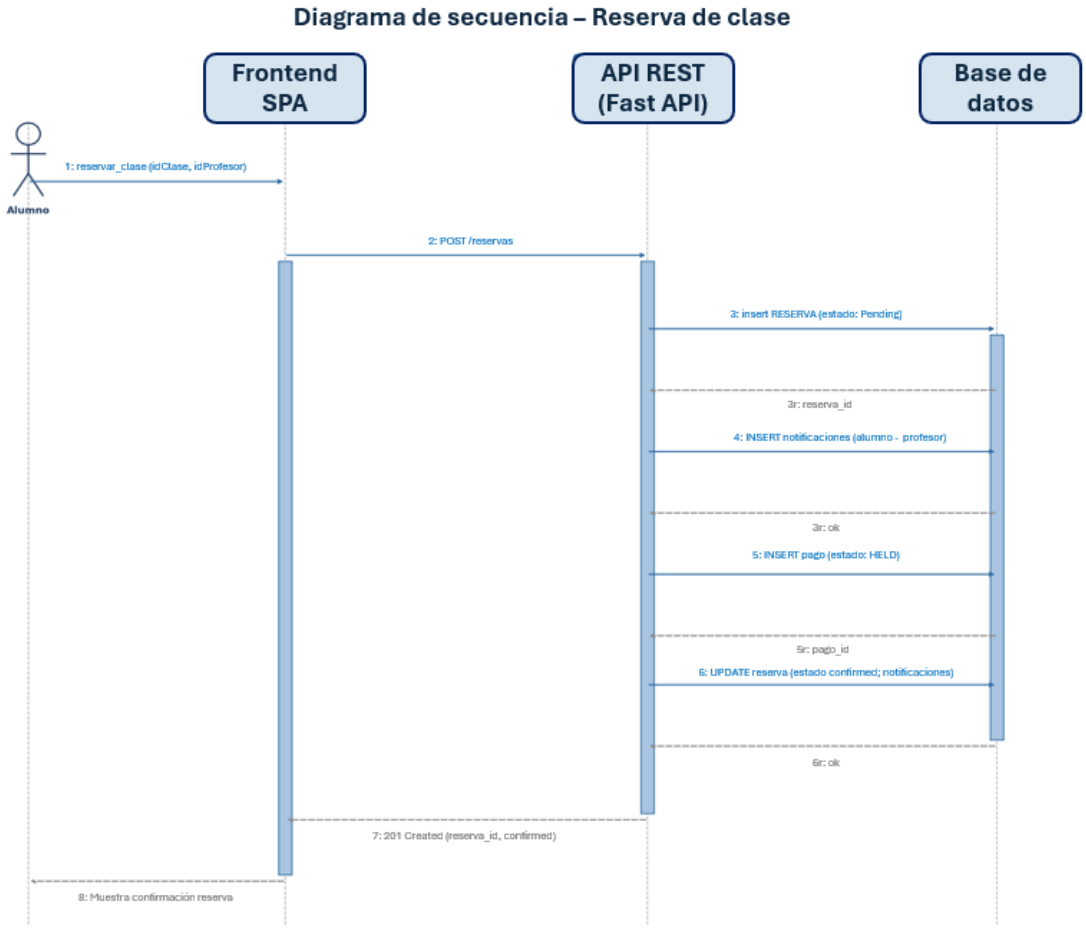

Ilustración 2: Diagrama de secuencia de flujo de reserva de clase

5. Conclusiones

PlayMatch demuestra que es posible diseñar e implementar, con un alcance de MVP, una plataforma que aporta orden, eficiencia y confianza a la relación entre alumnos y profesores de pádel. El sistema desarrollado es funcional, validado y desplegable, y constituye una base sólida para su evolución hacia un producto en producción. Las limitaciones del MVP —pasarela de pago simulada, cobertura geográfica reducida, ausencia de panel de administración— están claramente delimitadas y son coherentes con el alcance del trabajo. El principal valor diferencial del proyecto reside en la combinación de un algoritmo de matching transparente, un modelo de confianza mediante escrow y un entrenador virtual determinísta que aporta valor sin dependencia de servicios externos.

6. Referencias

[1] Sinnott, R. W. (1984). Virtues of the Haversine. Sky and Telescope, 68(2), 159.

[2] FIP. (2025). World Padel Report 2025. https://www.padelfip.com

[3] Playtomic. (2025). Global Padel Report 2025. https://playtomic.com

[4] OWASP Foundation. (2021). OWASP Top 10. https://owasp.org/Top10/

[5] IETF. (2015). RFC 7519: JSON Web Token (JWT).

[6] Ramírez, S. FastAPI Documentation. https://fastapi.tiangolo.com

PLAYMATCH: DESIGN AND IMPLEMENTATION OF A WEB APPLICATION FOR CONNECTING PADEL COACHES AND STUDENTS

Author: Rotaeche Montero, Amaia.
Supervisor: Férnadez-Pacheco Sánchez-Migallón, Atilano Ramiro
Collaborating entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

PlayMatch is a web platform for booking padel classes, developed as a Master's thesis. The backend is built with Python/FastAPI and a relational database, the frontend with React, and production deployment uses Docker Compose. The thesis documents the full development cycle: requirements analysis, UML design, REST API implementation, testing with pytest (78% coverage), and project planning.

Keywords: padel, web application, matching algorithm, FastAPI, React, escrow, minimum viable product.

1. Introduction

Padel has become one of the fastest-growing sports in Spain, exceeding 35 million players worldwide in 2025 (FIP, 2025) with more than 109,000 registered licences in Spain. This growth has multiplied the demand for private lessons taught by self-employed coaches. However, the relationship between students and coaches is still managed through informal channels and scattered tools, creating operational friction and distrust in a financial transaction normally closed between strangers.

2. Project definition

This Master's Thesis addresses this problem through the design and implementation of PlayMatch, a full-stack web application that structures and digitalises that relationship, centralising supply and demand and supporting the complete cycle of a private lesson: matching, booking, escrow payment, communication and rating.

3. System Description

PlayMatch was built as a full-stack application with a FastAPI backend on a relational database managed with SQLAlchemy and Alembic, and a single-page frontend in React and TypeScript. The technical core is a matching algorithm that works in two stages: (1) filtering by city, availability and validity, and (2) a scoring function that combines

geographic distance computed using the Haversine formula (Sinnott, 1984), level difference and a price penalty, producing a transparent and reproducible recommendation. The system also implements an escrow payment cycle, multi-dimensional ratings, dispute management and a deterministic virtual coach.

4. Results

The result of the work is a functional and validated prototype integrating all planned modules. Testing with pytest achieved 78 % global coverage, with critical modules (auth: 91 %, matching: 88 %) above average. Functional validation verified the main workflows for all three roles (student, coach, administrator). The system is fully containerised with Docker Compose and deployable in a reproducible manner.

5. Conclusions

PlayMatch demonstrates that it is possible to design and implement, within an MVP scope, a platform that brings order, efficiency and trust to the relationship between padel students and coaches. The developed system is functional, validated and deployable, and constitutes a solid foundation for evolution towards a production product. The MVP limitations — simulated payment gateway, limited geographic coverage, no administration panel — are clearly defined and coherent with the scope of the work. The main differential value lies in the combination of a transparent matching algorithm, a trust model based on escrow and a deterministic virtual coach that adds value without dependence on external services.

6. References

[1] Sinnott, R. W. (1984). Virtues of the Haversine. Sky and Telescope, 68(2), 159.

[2] FIP. (2025). World Padel Report 2025. https://www.padelfip.com

[3] Playtomic. (2025). Global Padel Report 2025. https://playtomic.com

[4] OWASP Foundation. (2021). OWASP Top 10. https://owasp.org/Top10/

[5] IETF. (2015). RFC 7519: JSON Web Token (JWT).

[6] Ramírez, S. FastAPI Documentation. https://fastapi.tiangolo.com

Índice de la memoria

Índice de ilustraciones

Índice de tablas

Capítulo 1. INTRODUCCIÓN

1.1 CONTEXTO Y MOTIVACIÓN

El pádel ha experimentado en la última década un crecimiento sostenido en España, visible tanto en el número de practicantes como en el volumen de instalaciones, la oferta formativa y su presencia social. Este crecimiento no se ha limitado a los clubes especializados, sino que se ha extendido a centros polivalentes, urbanizaciones y entornos de práctica descentralizados, lo que ha dado lugar a un mercado más capilar y heterogéneo que el de otras disciplinas. En este escenario ha cobrado protagonismo la figura del entrenador que trabaja de forma autónoma o semiautónoma, desplazándose entre distintas pistas y gestionando su cartera de alumnos por cuenta propia.

Sin embargo, la relación entre alumno y profesor sigue apoyándose, en muchos casos, en canales informales y en herramientas que no están conectadas entre sí. Localizar a un profesor adecuado, comprobar cuándo tiene hueco, acordar el precio, reservar la clase o pagarla son tareas que suelen resolverse a través de la mensajería instantánea o del boca a boca. Esta dispersión genera fricción en el día a día, dificulta el seguimiento de la actividad y, sobre todo, introduce desconfianza en una operación económica que se cierra con frecuencia entre personas que no se conocen.

De esta situación surge PlayMatch, una aplicación concebida para ordenar y digitalizar dicha relación. El modelo de negocio de la plataforma fue objeto de un estudio previo, centrado en su viabilidad estratégica y económica. Este Trabajo Fin de Máster parte de aquel planteamiento, pero dirige la atención hacia la dimensión técnica del proyecto: el diseño y la construcción de una aplicación realmente funcional que dé forma a la propuesta y que permita aplicar, de manera integrada, las competencias propias de la ingeniería.

1.2 PLANTEAMIENTO DEL PROBLEMA

El problema de partida es la falta de una solución integrada que cubra, de forma coordinada, los distintos pasos que componen la relación entre alumno y profesor en el pádel. Hoy, el descubrimiento de profesores, la consulta de la disponibilidad, la contratación, el

pago y el seguimiento posterior se reparten entre canales diversos y poco formalizados, con la consiguiente pérdida de eficiencia y de confianza.

En el fondo, se trata de un problema de emparejamiento. La plataforma debe conectar a cada alumno con los profesores que mejor se ajustan a lo que busca, teniendo en cuenta a la vez la ubicación, el nivel de juego, el precio y la disponibilidad horaria. Pero el emparejamiento es solo el principio: una vez que las dos partes entran en contacto, el sistema tiene que sostener con garantías todo el ciclo de la clase, que abarca la reserva, el pago, la impartición de la sesión, su valoración y la resolución de las incidencias que puedan surgir.

La confianza es, en este contexto, un elemento crítico. Como el pago se produce por adelantado y, a menudo, entre desconocidos, la solución debe proteger por igual al alumno y al profesor. PlayMatch aborda esta cuestión con una lógica de pago inspirada en plataformas de intermediación ya consolidadas: el importe abonado por el alumno queda retenido y no llega al profesor hasta que la clase se ha impartido o hasta que se resuelve una eventual disputa.

1.3 PRESENTACIÓN DE PLAYMATCH

PlayMatch es una plataforma digital, accesible desde un navegador web, cuyo fin es poner en contacto a los jugadores de pádel que buscan clases con los profesores que las imparten. Su propósito es reunir en un mismo lugar la oferta y la demanda de clases particulares y acompañar todo el recorrido de la relación formativa, desde que el alumno encuentra a un profesor adecuado hasta que valora la sesión, pasando por la reserva, el pago y la comunicación entre ambos. En la plataforma conviven tres perfiles de usuario, el alumno, el profesor y el administrador, cada uno con sus propias funciones.

El rasgo que distingue a PlayMatch es su mecanismo de recomendación, o matching, que propone al alumno las clases disponibles que mejor encajan con su perfil a partir de criterios como la ubicación, el nivel, el precio y el horario. Alrededor de ese núcleo se sitúa el resto de funcionalidades, que dan cuerpo a la operativa de la plataforma: la publicación de la disponibilidad por parte de los profesores, la reserva de clases con un esquema de pago que protege a ambas partes, la mensajería interna, las valoraciones, la gestión de incidencias y un entrenador virtual que ofrece orientación técnica al alumno.

El trabajo se centra en construir una primera versión funcional de la plataforma, planteada como producto mínimo viable. Cómo se ha organizado la arquitectura, qué tecnologías se han empleado y de qué modo se ha implementado cada uno de estos módulos son cuestiones que se abordan en los capítulos siguientes; en esta introducción simplemente se presenta la propuesta y se sitúa su alcance.

1.4 DATOS TRATADOS Y NECESIDAD DE SEGURIDAD

Por la propia naturaleza del servicio, PlayMatch maneja información personal y, en parte, sensible: datos de identidad de los usuarios, información sobre su ubicación, las conversaciones que mantienen alumnos y profesores y los datos ligados a las operaciones económicas que se canalizan a través de la plataforma. Trabajar con este tipo de información sitúa a la seguridad y a la protección de datos en un primer plano dentro del proyecto.

Por esa razón, la memoria reserva un capítulo (5.4) específico a la seguridad y a la protección de los datos sensibles, donde se explican las medidas adoptadas para preservar la confidencialidad, controlar el acceso a la información y reducir al mínimo los datos tratados, además de una reflexión sobre el cumplimiento del Reglamento General de Protección de Datos. Aquí interesa únicamente dejar constancia de que la seguridad no se añadió a posteriori, sino que estuvo presente desde las primeras decisiones de diseño.

1.5 ESTRUCTURA DE LA MEMORIA

La memoria se organiza en ocho capítulos, a los que se añaden la bibliografía y los anexos, siguiendo el recorrido del proyecto desde su fundamentación hasta la valoración de los resultados. Tras esta introducción, el capítulo 2 describe las tecnologías empleadas en el desarrollo, presentando el marco técnico sobre el que se construye la solución. El capítulo 3 revisa el estado de la cuestión, situando la propuesta frente a las plataformas existentes y razonando las decisiones de diseño adoptadas.

El capítulo 4 define el trabajo en su conjunto: recoge la justificación del proyecto, los objetivos y el alcance del producto mínimo viable, la metodología de desarrollo seguida y la planificación temporal junto con la estimación económica. El capítulo 5, el más extenso, expone el sistema desarrollado; abarca el análisis de requisitos, el diseño de la arquitectura

y el modelo de dominio, el esquema de la base de datos y los diagramas UML, el algoritmo de matching y la gestión de la ubicación, la seguridad y la protección de datos, las reglas de negocio y el entrenador virtual, y finalmente la implementación del backend y del frontend y el despliegue en los distintos entornos de ejecución.

El capítulo 6 presenta el análisis de resultados, con la estrategia de pruebas y la validación del sistema. El capítulo 7 recoge las conclusiones del trabajo y las líneas de desarrollo futuro. Cierran el documento la bibliografía (capítulo 8) y los anexos, entre los que se incluyen los manuales de instalación y usuario. En conjunto, esta estructura busca mantener una correspondencia clara entre el problema planteado, la solución propuesta y la evidencia de su validación.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

La selección de tecnologías se ha guiado por su adecuación al problema y no por una preferencia previa por herramientas concretas. Se han valorado la mantenibilidad del código, la rapidez razonable para levantar un MVP funcional, la madurez de cada ecosistema y la disponibilidad de documentación y comunidad. El resultado es una arquitectura con una separación clara entre la interfaz, la lógica de negocio y la persistencia, que reduce el acoplamiento y facilita la evolución del sistema. A continuación se describen las tecnologías empleadas en cada capa.

2.1 BACKEND: FASTAPI, PYDANTIC Y OPENAPI

En el lado del servidor se ha empleado FastAPI, un framework de Python orientado a la construcción de API REST. Su elección responde a tres motivos: ofrece un rendimiento elevado gracias a su naturaleza asíncrona, acelera el desarrollo al apoyarse en las anotaciones de tipo del propio lenguaje y genera de forma automática la documentación de la API. La validación y la serialización de los datos se apoyan en Pydantic, que permite definir mediante tipos los esquemas de entrada y salida y garantiza que la información que entra en el sistema cumple el formato esperado. Además, FastAPI produce automáticamente una especificación OpenAPI y una documentación interactiva, accesible a través de Swagger UI y ReDoc, muy útil tanto para probar la API durante el desarrollo como para documentarla.

2.2 PERSISTENCIA: SQLALCHEMY Y ALEMBIC

Para la persistencia de los datos se ha utilizado SQLAlchemy en su versión 2, un mapeador objeto-relacional que traduce las clases de Python a tablas de una base de datos relacional y abstrae buena parte del SQL, lo que facilita además la portabilidad entre distintos motores. La evolución del esquema se gestiona con Alembic, que permite versionar los cambios en la estructura de la base de datos mediante migraciones, de forma análoga a como Git versiona el código. Por defecto, el sistema trabaja con SQLite, una base de datos ligera y sin servidor que resulta idónea para el desarrollo y para la demostración del MVP; la

configuración deja abierta la puerta a un motor más robusto, como PostgreSQL, en un eventual despliegue en producción.

2.3 AUTENTICACIÓN Y SEGURIDAD WEB

Cualquier aplicación web que gestione datos personales debe atender a la autenticación de los usuarios, a la autorización de las acciones que cada uno puede realizar, a la confidencialidad de las comunicaciones y a la validación de las entradas, sin perder de vista las vulnerabilidades más habituales recogidas en referencias como el OWASP Top 10. PlayMatch aborda estas cuestiones combinando el flujo OAuth2 de tipo password con tokens JWT para una autenticación sin estado, y el algoritmo bcrypt para almacenar las contraseñas de forma cifrada. Dada su importancia en este proyecto, la seguridad y la protección de los datos sensibles se tratan en detalle en el capítulo 5.4.

2.4 FRONTEND: REACT, TYPESCRIPT, VITE Y TAILWIND

La interfaz de usuario se ha construido como una aplicación de página única con React, una biblioteca que organiza la interfaz en componentes reutilizables y facilita la gestión del estado. Sobre React se ha empleado TypeScript, que añade tipado estático a JavaScript y ayuda a detectar errores antes de la ejecución, mejorando la mantenibilidad del código. La herramienta Vite se encarga de la construcción del proyecto y del servidor de desarrollo, con tiempos de recarga muy rápidos, mientras que Tailwind CSS aporta un sistema de estilos basado en utilidades que permite mantener una estética coherente sin escribir hojas de estilo extensas. La navegación entre las distintas vistas se resuelve con React Router.

2.5 GEOLOCALIZACIÓN: NOMINATIM Y CATÁLOGO DE DISTRITOS

El mecanismo de recomendación necesita coordenadas geográficas para estimar las distancias entre alumno y profesor. Para ello se han combinado dos enfoques. El principal es un catálogo propio de ciudades y distritos con coordenadas aproximadas de cada zona, que evita depender de servicios externos en el uso habitual de la aplicación. De forma complementaria, se ofrece la posibilidad de geocodificar texto libre mediante Nominatim, el

servicio de geocodificación de OpenStreetMap, al que se accede a través de la biblioteca httpx. Este planteamiento se desarrolla con más detalle en el capítulo 5.3 dedicado al algoritmo de matching.

2.6 HERRAMIENTAS DE DESARROLLO Y DESPLIEGUE

El desarrollo se ha apoyado en un conjunto de herramientas habituales en la ingeniería del software. El control de versiones se ha llevado con Git; las dependencias de Python se han aislado en un entorno virtual; y la contenedorización mediante Docker y Docker Compose permite reproducir el entorno de ejecución de forma consistente, lo que simplifica tanto las pruebas como un futuro despliegue. La documentación interactiva generada con OpenAPI ha servido, además, como banco de pruebas de la API durante el desarrollo. La automatización del despliegue mediante integración continua queda planteada como línea de trabajo futuro.

Capítulo 3. ESTADO DE LA CUESTIÓN

Este capítulo cumple una doble función. Por un lado, sitúa a PlayMatch dentro del ecosistema digital del deporte, repasando las soluciones que ya existen y detectando los huecos que justifican una propuesta especializada. Por otro, presenta el marco tecnológico sobre el que se ha construido la aplicación, explicando qué herramientas se han elegido y con qué criterio. No se entra todavía en los detalles de implementación, que se reservan para capítulos posteriores; aquí interesa ofrecer el contexto necesario para entender las decisiones de diseño.

3.1 SOLUCIONES EXISTENTES EN EL ÁMBITO DEPORTIVO

El mercado actual cuenta con aplicaciones que han contribuido de forma notable a la digitalización del deporte, sobre todo en lo relativo a la reserva de pistas y a la organización de partidos. El ejemplo más representativo en el ámbito de los deportes de raqueta es Playtomic, una plataforma ampliamente implantada que permite localizar pistas, reservarlas y organizar partidos entre jugadores de nivel similar. Su impacto en la ocupación de las instalaciones y en la visibilidad de los clubes es indudable, y ha servido para profesionalizar buena parte de la gestión operativa del sector.

Según el informe Global Padel Report 2025 elaborado por Playtomic, la plataforma conecta a más de 4 millones de usuarios registrados con más de 6.000 clubes afiliados en 63 países, y cerró en 2025 una ronda de financiación de 65 millones de euros para impulsar su expansión internacional. Smash, por su parte, opera principalmente en el mercado hispano y combina reserva de pistas con búsqueda de compañeros de partido, mientras que ClassPass ofrece un modelo de suscripción multideporte orientado a clases en estudios y gimnasios. Ninguna de estas soluciones aborda específicamente la relación formativa entre alumno y entrenador autónomo de pádel fuera de los circuitos de clubes y academias, que es precisamente el nicho que PlayMatch pretende cubrir (Playtomic, 2025).

Ahora bien, la propuesta de valor de estas plataformas se orienta a la capa transaccional y social del deporte recreativo, no al ciclo formativo completo de la relación entre alumno y profesor. Existen, además, aplicaciones centradas en el seguimiento físico o

en la distribución de contenidos de entrenamiento que resultan útiles en ámbitos concretos, pero que rara vez integran la comunicación docente, la planificación de las sesiones y el seguimiento de la progresión técnica. Como resultado, quien quiere dar o recibir clases termina combinando varias herramientas parciales para cubrir un mismo proceso.

Entre las plataformas con mayor presencia en el mercado del pádel destacan Playtomic, que conecta a más de 4 millones de usuarios con más de 6.000 clubes en 63 países y cuenta con una creciente integración de entrenadores en su ecosistema; y, en un segmento más especializado, Coach My Padel, The Padel School, PadelGlobo y ToPadel, que ofrecen servicios de coaching online, formación bajo demanda o intermediación puntual en cursos y camps. Ninguna de ellas, sin embargo, integra el ciclo completo de la clase presencial particular: emparejamiento, reserva, pago seguro y comunicación en un único flujo.

Para el diseño del mecanismo de recomendación conviene mirar también a otros sectores. Las plataformas de citas, por ejemplo, han consolidado una lógica de emparejamiento basada en la afinidad, la proximidad geográfica y las preferencias declaradas. Aunque su objetivo y su contexto son muy distintos, aportan patrones técnicos perfectamente trasladables, como la ponderación de varios criterios, la priorización de resultados o la mejora iterativa de las recomendaciones. PlayMatch se inspira en esos principios para conectar a alumnos y profesores con mayor probabilidad de ajuste, pero manteniendo el foco en la utilidad formativa.

Algo parecido ocurre con el modelo de pago. Plataformas de compraventa de segunda mano como Vinted o Wallapop han popularizado un esquema de intermediación en el que el dinero queda retenido hasta que la operación se completa, lo que protege tanto al comprador como al vendedor. Trasladado a las clases de pádel, este enfoque permite asegurar el pago en el momento de la reserva y liberarlo al profesor solo cuando la clase se ha impartido, reduciendo así la desconfianza y las fricciones asociadas a cancelaciones o incidencias.

3.2 LIMITACIONES DETECTADAS Y OPORTUNIDAD PARA PLAYMATCH

Del repaso anterior se desprende que, pese a los avances en la digitalización deportiva, persisten limitaciones cuando el problema se mira desde la enseñanza personalizada. La más evidente es la escasa diferenciación entre perfiles: las necesidades de un alumno y las de un profesor no siempre se traducen en flujos de uso adaptados a cada rol. A ello se suma una integración insuficiente entre comunicación, planificación y registro de la actividad, que obliga a apoyarse en herramientas externas y debilita la trazabilidad del proceso formativo.

Esta situación abre una oportunidad clara para una solución especializada. En lugar de competir de frente con las plataformas consolidadas en la reserva de pistas o en la organización social del juego, PlayMatch se posiciona en el espacio menos cubierto: la relación entre alumno y profesor en torno al proceso de aprendizaje. Su valor diferencial está en reunir en una misma experiencia el descubrimiento del profesor, el emparejamiento, la reserva, el pago, la comunicación y la valoración, áreas que hoy aparecen dispersas entre distintas aplicaciones.

3.3 CRITERIOS DE SELECCIÓN Y SÍNTESIS

En conjunto, el marco tecnológico elegido busca un equilibrio entre productividad, mantenibilidad y separación de responsabilidades. Se trata de tecnologías maduras, ampliamente documentadas y respaldadas por comunidades activas, lo que reduce el riesgo técnico y facilita encontrar soluciones a los problemas que surgen durante el desarrollo. La combinación de un backend en Python con FastAPI y de un frontend en React con TypeScript es, además, una pareja consolidada en el desarrollo web actual, lo que aporta garantías de continuidad.

Estas decisiones sientan las bases del trabajo que se describe en los capítulos siguientes. Una vez presentados el contexto y el marco tecnológico, el capítulo 4 aborda la definición del trabajo: justifica el proyecto desde una perspectiva técnica y de mercado, formula los objetivos y el alcance del producto mínimo viable, describe la metodología de desarrollo adoptada y cierra con la planificación temporal y la estimación económica.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

El pádel es el deporte de mayor crecimiento en España y uno de los más practicados de Europa: con más de cuatro millones de jugadores y veinte mil pistas instaladas, su popularidad ha dado lugar a una demanda creciente de formación personalizada. Sin embargo, el mercado de clases particulares de pádel sigue siendo opaco y fragmentado: no existe ninguna plataforma digital que conecte de forma estructurada y segura a alumnos con profesores cualificados. PlayMatch nace para cubrir ese hueco con una propuesta técnicamente diferenciada y con un modelo de negocio sostenible.

4.1.1 OPORTUNIDAD DE MERCADO

Las plataformas digitales de pádel existentes —Playtomic, Padelmanager— se centran en la reserva de pistas y la organización de partidos recreativos. Ninguna ofrece un mercado estructurado para la contratación de clases particulares. El alumno que quiere mejorar técnicamente recurre hoy a canales informales: grupos de WhatsApp, recomendaciones de boca a boca o publicaciones en redes sociales. Esto genera incertidumbre en precio, calidad y compromiso, y deja sin monetizar un segmento con una demanda latente de cientos de miles de usuarios solo en España. A nivel europeo, el pádel tiene presencia federada en más de treinta países, lo que abre una vía de expansión natural para una plataforma que haya validado el modelo en el mercado doméstico.

4.1.2 PROPUESTA DE VALOR DIFERENCIAL

PlayMatch no es una plataforma de reserva de pistas: es un mercado de talento deportivo. Tres elementos la diferencian de cualquier solución existente en el mercado:

Algoritmo de matching inteligente: combina proximidad geográfica, nivel de juego, disponibilidad horaria y valoraciones históricas para conectar al alumno con el profesor más adecuado. No es una lista de resultados genérica, sino una recomendación personalizada que reduce la fricción del proceso de búsqueda y aumenta la probabilidad de que la clase se materialice.

Pagos con retención (escrow): el importe de la clase queda bloqueado en el monedero de la plataforma hasta que esta se imparte, protegiendo al alumno frente a cancelaciones unilaterales y garantizando al profesor el cobro efectivo si cumple con lo acordado. Este modelo, popularizado por plataformas de comercio electrónico como Wallapop o Vinted, aporta la confianza necesaria para que dos desconocidos cierren una transacción económica.

Entrenador virtual: un módulo de recomendación personalizada que sugiere ejercicios técnicos, contenido audiovisual y material deportivo adaptados al perfil y nivel del alumno. Genera valor más allá de la reserva puntual, crea un vínculo recurrente con la plataforma y abre una vía de monetización adicional mediante acuerdos con fabricantes de material deportivo.

4.1.3 VIABILIDAD TÉCNICA Y MODELO DE NEGOCIO

PlayMatch es técnicamente viable como MVP en un horizonte de pocos meses con un equipo reducido, gracias a la madurez del ecosistema tecnológico elegido. Su modelo de negocio se sustenta en una comisión sobre cada transacción completada, de forma análoga al modelo marketplace de Airbnb o Glovo: los ingresos crecen directamente con el volumen de reservas sin requerir grandes inversiones en infraestructura propia. El modelo presenta efectos de red positivos: cada nuevo profesor que se incorpora amplía el catálogo disponible para los alumnos, y cada nuevo alumno aumenta la demanda para los profesores, haciendo la plataforma más valiosa para ambos colectivos a medida que crece.

4.2 OBJETIVOS

Presentado ya el contexto del proyecto y la propuesta de PlayMatch, este apartado concreta los objetivos que orientan el trabajo y delimita su alcance. Se diferencia entre un objetivo general, que resume la finalidad última, y un conjunto de objetivos específicos que detallan las metas necesarias para alcanzarla. El capítulo termina precisando qué incluye el producto mínimo viable y qué limitaciones se han asumido de forma deliberada.

4.2.1 OBJETIVO GENERAL

El objetivo general de este Trabajo Fin de Máster es diseñar e implementar una aplicación web funcional, PlayMatch, capaz de ordenar y digitalizar la relación entre

alumnos y profesores de pádel. La aplicación debe conectar a ambos perfiles de manera ágil y acompañar el ciclo completo de la clase particular, desde la búsqueda del profesor hasta la valoración del servicio, sobre una base técnica sólida, mantenible y abierta a una evolución posterior.

Este objetivo se concreta en un producto mínimo viable que dé forma a la propuesta de valor de la plataforma y que sirva, a la vez, como ejercicio de ingeniería en el que confluyen varias competencias del Máster: el análisis de requisitos, el diseño de la arquitectura y del modelo de datos, el desarrollo de los componentes de servidor y de cliente, la incorporación de mecanismos de seguridad y la validación del sistema.

4.2.2 OBJETIVOS ESPECÍFICOS

Para alcanzar ese objetivo general, el trabajo se desglosa en una serie de objetivos específicos que recorren todo el ciclo de desarrollo. El primero consiste en analizar las necesidades de los perfiles implicados y traducirlas en un marco coherente de requisitos, tanto funcionales como no funcionales. El segundo, en diseñar una arquitectura modular que separe con claridad la interfaz, la lógica de negocio y la persistencia, junto con un modelo de datos que dé soporte a las distintas funcionalidades.

A partir de ahí se sitúan los objetivos relacionados con las capacidades centrales del producto: desarrollar el mecanismo de recomendación que proponga al alumno las clases más adecuadas según la ubicación, el nivel, el precio y la disponibilidad; construir el ciclo de reservas sobre un esquema de pago retenido que ofrezca garantías a las dos partes y deje traza de cada operación; y añadir las funciones de comunicación, valoración y gestión de incidencias que dan continuidad a la relación entre alumno y profesor. A ellos se suma el desarrollo de un entrenador virtual que proporcione al alumno una orientación técnica básica.

Por último, el trabajo asume dos objetivos de carácter transversal. Por un lado, proteger los datos personales que maneja la plataforma mediante la autenticación de los usuarios, el control de acceso por rol y por recurso y la minimización de la información almacenada. Por otro, validar el funcionamiento del sistema, comprobando que las funcionalidades responden a lo esperado y respetan las reglas de negocio establecidas.

4.2.3 ALCANCE DEL MVP Y LIMITACIONES

El alcance del proyecto abarca el ciclo completo de análisis, diseño e implementación de una primera versión operativa de la plataforma, entendida como producto mínimo viable. Esa versión se concentra en los procesos esenciales de la relación entre alumno y profesor y en levantar una base funcional sólida sobre la que crecer en etapas posteriores. El resultado es una aplicación web completa, con sus partes de servidor y de cliente, que permite registrar usuarios, gestionar perfiles y disponibilidad, recomendar y reservar clases, simular el pago y su liquidación, comunicarse, valorar y tramitar incidencias.

Coherentemente con esa delimitación, hay capacidades propias de un producto comercial maduro que quedan fuera de esta etapa o se resuelven de forma simulada. El pago es una simulación, de modo que no se procesan cobros reales ni se integran proveedores externos; el rol de administrador se gestiona a mano sobre la base de datos, sin un panel propio; la cobertura geográfica se restringe a un conjunto acotado de ciudades; el entrenador virtual se apoya en un catálogo estático y en un motor de reglas, sin recurrir a servicios externos de inteligencia artificial; y el sistema todavía no cuenta con despliegue público ni con integración continua.

Estas decisiones no responden a una carencia, sino a un criterio de viabilidad técnica y académica: acotar el desarrollo permite ganar profundidad en el núcleo funcional, mantener la trazabilidad de los resultados y cuidar la calidad de lo implementado. Las limitaciones señaladas no restan valor demostrativo al producto mínimo viable y, de hecho, perfilan una hoja de ruta para futuras iteraciones que se retoma en el capítulo de conclusiones.

4.3 METODOLOGÍA DE DESARROLLO

El desarrollo de PlayMatch ha seguido un enfoque iterativo e incremental, habitual en proyectos de software en los que los requisitos se van perfilando a medida que avanza la construcción. En lugar de planificar todo el sistema de una sola vez, el trabajo se ha organizado en ciclos cortos que combinan análisis, diseño, implementación y verificación, y en los que cada bloque funcional se aborda según el valor que aporta al caso de uso.

En la práctica, se partió de los cimientos del sistema y se fue avanzando hacia las funcionalidades de mayor valor para el usuario, reservando en cada iteración un margen para comprobar tanto el comportamiento de los componentes como su ajuste a los objetivos. Esta forma de trabajar ayuda a detectar pronto los problemas y mantiene una línea clara entre lo que se necesitaba, lo que se decidió y lo que finalmente se obtuvo.

El detalle de la metodología, las herramientas utilizadas y la organización concreta del trabajo se recogen en los capítulos dedicados a la implementación y al despliegue del sistema.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

Este capítulo analiza el coste real de desarrollo de PlayMatch y la planificación temporal del proyecto. Se distinguen dos perspectivas: la inversión efectiva durante el desarrollo académico —prácticamente nula gracias al uso de herramientas gratuitas o de uso académico— y la inversión necesaria para una eventual explotación comercial de la plataforma.

4.4.1 INVERSIÓN EN DESARROLLO

El desarrollo de PlayMatch se ha llevado a cabo con herramientas que no han supuesto coste económico directo. Todas las tecnologías utilizadas —tanto del stack de backend como del frontend— son de código abierto o disponen de planes gratuitos orientados al uso académico y personal. La Tabla 1 recoge el inventario completo de herramientas empleadas y su coste asociado.

Tabla 1: Herramientas y tecnologías empleadas en el desarrollo de PlayMatch y su coste asociado.

Herramienta / Tecnología	Plan utilizado	Coste
Python, FastAPI, PostgreSQL, Redis	Código abierto (licencia MIT / Apache)	0 €
React, Vite, Tailwind CSS	Código abierto	0 €
GitHub (repositorio + CI/CD)	Free / Student Developer Pack	0 €
VS Code + GitHub Copilot	Education Pack (licencia académica)	0 €
Docker Desktop	Personal / Educational	0 €
Figma (diseño UI)	Starter (gratuito)	0 €
Entorno de despliegue (local)	Equipo propio	0 €
TOTAL		**0 €**

En consecuencia, la inversión directa en herramientas durante el desarrollo académico ha sido de 0 €. El único coste no monetizado es el tiempo de desarrollo: aproximadamente 400 horas distribuidas a lo largo de seis meses (enero–junio de 2026), equivalentes a un coste de oportunidad de entre 14.000 y 20.000 € a tarifa de mercado junior–sénior (35–50 €/h).

4.4.2 PLANIFICACIÓN: DIAGRAMA DE GANTT

El proyecto se ha organizado en seis fases iterativas siguiendo una metodología incremental. La Ilustración 3 recoge la planificación temporal con las fechas de inicio y fin de cada fase y las horas dedicadas. El diagrama de Gantt correspondiente se adjunta como material complementario en formato Excel.

Diagrama de Gantt — PlayMatch TFM (Enero – Junio 2026)															
Fase	Inicio	Fin	Horas	Actividades principales	Enero		Febrero		Marzo		Abril		Mayo		Junio	
					S1	S2	S1	S2	S1	S2	S1	S2	S1	S2	S1	S2
Fase 1: Análisis y Requisitos	01/01/2026	31/01/2026	50	Casos de uso, requisitos func. y no func., modelo de dominio												
Fase 2: Diseño del Sistema	01/02/2026	28/02/2026	60	Arquitectura, modelo de datos, diseño UI/UX en Figma												
Fase 3: Impl. Backend	01/03/2026	31/03/2026	90	API REST FastAPI, auth JWT, reservas, pagos escrow, chat												
Fase 4: Frontend y Matching	01/04/2026	30/04/2026	90	SPA React/TypeScript, motor de matching, entrenador virtual												
Fase 5: Pruebas e Integración	01/05/2026	31/05/2026	60	pytest, cobertura, pruebas de integración y carga												
Fase 6: Documentación y Deploy	01/06/2026	30/06/2026	50	Memoria TFM, manuales de usuario e instalación, despliegue												
TOTAL			400													

Ilustración 3: Planificación temporal en formato Gantt Chart

4.4.3 GO TO MARKET: INVERSIÓN PARA LA EXPLOTACIÓN COMERCIAL

PlayMatch, en su estado actual, es un prototipo funcional que demuestra la viabilidad técnica de la plataforma. Si en el futuro se desea lanzar el producto al mercado, será necesario acometer una inversión significativa en varias dimensiones. A continuación se describen las principales áreas de inversión identificadas, tomando como referencia el análisis de viabilidad económica del TFG de origen (PlayMatch, 2024).

- **Mejoras técnicas post-MVP.** El prototipo cubre los flujos principales pero no está optimizado para escala. Se necesitaría inversión en: refactorización para alta concurrencia, integración de un gateway de pagos real (Stripe / Redsys), sistema de notificaciones push, app móvil nativa (iOS y Android) y mecanismos de geolocalización avanzada.
- **Infraestructura en la nube.** El despliegue en producción en plataformas cloud (AWS, GCP o Azure) supondría un coste mensual recurrente estimado entre 300 y 800 €/mes en función del volumen de usuarios, incluyendo compute (EC2 / Cloud Run), base de datos gestionada (RDS / Cloud SQL), almacenamiento de archivos (S3), CDN y servicio de cola de mensajería.
- **Marketing y adquisición de usuarios.** El modelo de negocio requiere masa crítica de alumnos y profesores. La estrategia de go-to-market incluiría SEO/SEM, presencia en redes sociales, acuerdos con clubs deportivos y centros de entrenamiento, y posiblemente publicidad en tiendas de aplicaciones (App Store, Google Play). La estimación del TFG de origen cifra el presupuesto de marketing inicial en 15.000–25.000 €.

- **Marco legal y cumplimiento normativo.** El lanzamiento como producto comercial exigiría: constitución de sociedad, registro de marca, redacción de términos y condiciones y política de privacidad conformes al RGPD, auditoría de protección de datos (DPO), y gestión del sistema de escrow de pagos bajo la normativa PSD2. Se estima un coste jurídico inicial de 5.000–10.000 €.
- **Equipo operativo.** A partir de cierto volumen de usuarios sería necesario contratar soporte al cliente, un community manager y, eventualmente, un segundo desarrollador para mantener el ritmo de mejoras. El coste de personal operativo mensual rondaría los 5.000–8.000 €.

4.4.4 SÍNTESIS

PlayMatch ha demostrado que es posible desarrollar un sistema de software completo y funcional con una inversión directa en herramientas prácticamente nula, aprovechando el ecosistema de código abierto y los programas académicos de los principales proveedores. El coste real del proyecto se concentra en el tiempo de desarrollo (~400 h), que a tarifa de mercado representaría entre 14.000 y 20.000 €. Para escalar PlayMatch a un producto comercial viable, la inversión estimada en el primer año rondaría los 80.000–120.000 €, incluyendo infraestructura, marketing, equipo y costes legales.

Capítulo 5. SISTEMA DESARROLLADO

Este capítulo describe el sistema PlayMatch en su conjunto, abarcando el análisis de requisitos, el diseño, los algoritmos, la seguridad, las reglas de negocio, la implementación y el despliegue. Para los detalles de instalación y configuración del entorno, véase el Anexo II: Manual de instalación y puesta en marcha.

5.1 ANÁLISIS DE REQUISITOS

El análisis de requisitos traduce las necesidades detectadas en los capítulos anteriores en un conjunto de especificaciones concretas que orientan el diseño y la implementación del sistema. En este capítulo se identifican primero los actores que interactúan con PlayMatch, se enumeran después los requisitos funcionales, no funcionales y de seguridad, y se recogen, por último, los casos de uso principales que se derivan de ellos.

5.1.1 IDENTIFICACIÓN DE ACTORES

- Visitante: usuario no autenticado que únicamente puede registrarse o iniciar sesión en la plataforma.
- Alumno: busca y reserva clases, efectúa el pago, valora al profesor y, si procede, abre una incidencia.
- Profesor: publica su disponibilidad, imparte las clases, atiende las reservas y responde a las incidencias.
- Administrador: rol interno que interviene de forma puntual para resolver las disputas que no se cierran entre las partes. A diferencia de alumnos y profesores, no se registra desde la interfaz sino que se gestiona directamente sobre la base de datos.

5.1.2 REQUISITOS FUNCIONALES

Los requisitos funcionales especifican qué debe permitir hacer el sistema. La *Tabla 2* los recoge identificados con la etiqueta RF y agrupados según el actor que los protagoniza, abarcando desde el registro y la autenticación hasta la gestión de incidencias por parte del administrador.

Tabla 2: Requisitos funcionales de PlayMatch.

ID	Actor	Requisito funcional
RF-01	Visitante	Registrarse como alumno o profesor indicando correo, nombre de usuario único y contraseña.
RF-02	Visitante	Iniciar sesión y obtener un token de autenticación para acceder a las funciones protegidas.
RF-03	Alumno	Crear y editar su perfil: nombre, ciudad, distrito, nivel y presupuesto máximo por clase.
RF-04	Alumno	Buscar clases mediante el algoritmo de matching, indicando el día y la franja horaria deseada.
RF-05	Alumno	Reservar una de las clases propuestas por el sistema.
RF-06	Alumno	Pagar la reserva de forma simulada, con una tarjeta nueva o con una guardada en el monedero.
RF-07	Alumno	Marcar la clase como completada, lo que libera el pago retenido al profesor.
RF-08	Alumno	Valorar al profesor tras la clase mediante varios criterios y un comentario opcional.
RF-09	Alumno	Abrir una incidencia una vez finalizada la clase.
RF-10	Alumno	Consultar el entrenador virtual para recibir consejos técnicos y recomendaciones de palas.
RF-11	Alumno	Gestionar el monedero, dando de alta o de baja métodos de pago.
RF-12	Profesor	Crear y editar su perfil profesional: titulación, descripción y tarifa por hora.
RF-13	Profesor	Publicar y gestionar sus franjas de disponibilidad.
RF-14	Profesor	Consultar y gestionar las reservas recibidas, incluida su cancelación.
RF-15	Profesor	Responder a las incidencias abiertas por el alumno dentro del plazo establecido.
RF-16	Profesor	Volver a publicar una franja liberada tras una cancelación.
RF-17	Profesor	Consultar el saldo acumulado por las clases impartidas.
RF-18	Alumno y profesor	Comunicarse mediante un chat asociado a cada reserva.
RF-19	Alumno y profesor	Recibir y consultar notificaciones sobre el estado de reservas, pagos e incidencias.
RF-20	Alumno y profesor	Consultar perfiles públicos, estadísticas y valoraciones de otros usuarios.
RF-21	Administrador	Resolver incidencias mediante reembolso total, liberación al profesor o reparto parcial.

5.1.3 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales describen las cualidades que debe satisfacer el sistema, más allá de las funciones concretas. La *Tabla 3* los recoge identificados con la etiqueta RNF, abarcando aspectos de usabilidad, rendimiento, seguridad, privacidad, mantenibilidad, portabilidad, fiabilidad y compatibilidad.

Tabla 3: Requisitos no funcionales de PlayMatch.

ID	Atributo	Descripción
RNF-01	Usabilidad	Interfaz clara y diferenciada por rol; el sistema guía al usuario para completar su perfil antes de operar.
RNF-02	Rendimiento	Tiempos de respuesta adecuados en las operaciones habituales de consulta y reserva.
RNF-03	Seguridad	Autenticación obligatoria en las operaciones sensibles y control de acceso por rol y por recurso.
RNF-04	Privacidad	Tratamiento responsable de los datos personales y minimización de los datos de pago.
RNF-05	Mantenibilidad	Arquitectura modular con separación entre interfaz, lógica de negocio y persistencia.
RNF-06	Portabilidad	Independencia del motor de base de datos y despliegue reproducible mediante contenedores.
RNF-07	Fiabilidad	Comportamiento estable y coherente de las reglas de negocio en escenarios de uso normal.
RNF-08	Compatibilidad	Acceso desde navegadores web modernos sin necesidad de instalar nada adicional.

5.1.4 REQUISITOS DE SEGURIDAD Y PRIVACIDAD

Dada la naturaleza de los datos que trata la plataforma, se han recogido de forma específica los requisitos de seguridad y privacidad, identificados con la etiqueta RS y resumidos en la *Tabla 4*. Constituyen un avance del capítulo 5.4 en el que se especificarán y detallarán las medidas implementadas para su cumplimiento.

Tabla 4: Requisitos de seguridad y privacidad (resumen).

ID	Requisito de seguridad y privacidad
RS-01	Las contraseñas se almacenan cifradas; en ningún momento se guardan ni se transmiten en claro.
RS-02	El acceso a las operaciones sensibles exige un token de autenticación válido y vigente.

ID	Requisito de seguridad y privacidad
RS-03	Cada usuario solo puede acceder a los recursos que le corresponden por su rol y por su participación en la reserva.
RS-04	De las tarjetas de pago solo se conservan datos parciales, nunca el número completo ni el código de seguridad.
RS-05	Las reglas de negocio de cancelación, retención y liberación de pagos se aplican íntegramente en el servidor.
RS-06	Las acciones relevantes quedan registradas mediante notificaciones y cambios de estado, dando trazabilidad al proceso.
RS-07	El tratamiento de datos personales se orienta a los principios del RGPD, con una finalidad limitada a la gestión de las clases.

5.1.5 CASOS DE USO

Los requisitos funcionales se traducen en un conjunto de casos de uso que describen las interacciones principales entre los actores y el sistema. La Ilustración 4 los resume gráficamente, indicando el actor que inicia cada uno y delimitando la frontera del sistema PlayMatch.

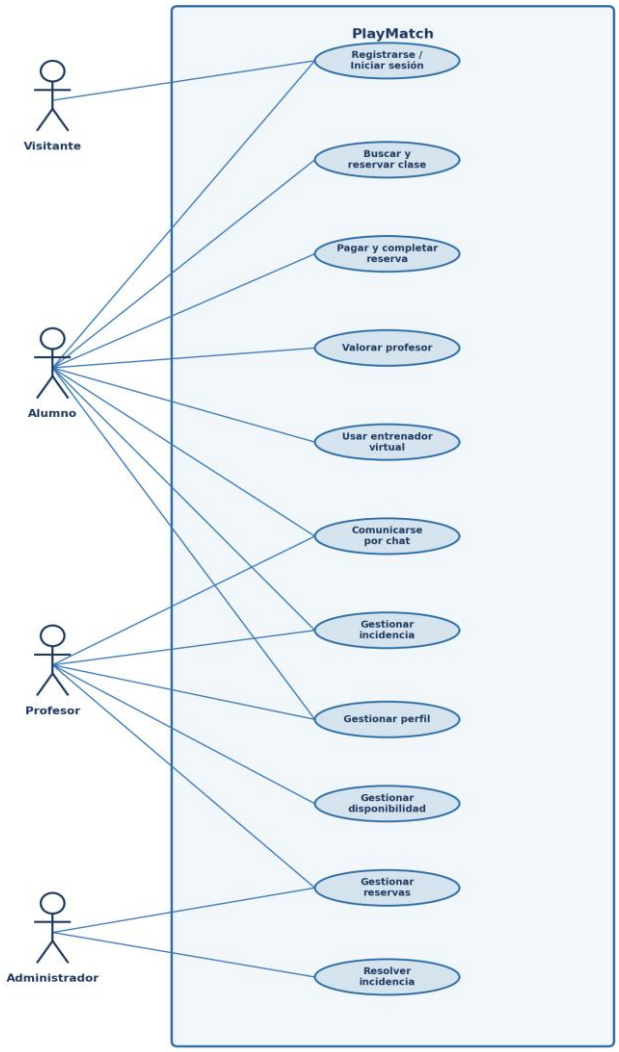

Ilustración 4: Diagrama de casos de uso de PlayMatch

El diagrama refleja que el alumno concentra la mayor parte de la actividad, al ser quien busca, reserva, paga y valora las clases, mientras que el profesor gestiona su oferta y atiende las reservas recibidas. Algunos casos de uso, como gestionar el perfil o comunicarse por chat, son compartidos por alumno y profesor. El administrador interviene de forma puntual, únicamente para resolver las incidencias que requieren arbitraje.

5.1.6 PRIORIZACIÓN Y CONCLUSIÓN

No todos los requisitos tienen la misma prioridad dentro del producto mínimo viable. Se han considerado prioritarios la autenticación, la gestión de perfiles, el matching, el ciclo de reserva y pago y la gestión de incidencias, por constituir el núcleo de la propuesta de valor. En un segundo plano se sitúan funcionalidades de apoyo como el entrenador virtual, las notificaciones o la consulta de perfiles públicos, que enriquecen la experiencia sin ser imprescindibles para el flujo principal. Esta priorización ha guiado el orden de la implementación y se refleja en el diseño del sistema, que se aborda en el capítulo siguiente.

5.2 DISEÑO DEL SISTEMA

Este capítulo describe el diseño de PlayMatch antes de entrar en los detalles de implementación. Se presenta primero la arquitectura general, después el modelo de dominio y el esquema relacional que sustentan la persistencia, y por último la visión dinámica del sistema a través del ciclo de vida de la reserva y de un diagrama de secuencia representativo. El objetivo es ofrecer una imagen coherente de cómo se organizan los componentes y cómo cooperan entre sí.

5.2.1 ARQUITECTURA GENERAL

PlayMatch sigue una arquitectura cliente-servidor de tres capas, con una separación nítida entre la presentación, la lógica de negocio y la persistencia. Esta organización, anticipada en el capítulo anterior, reduce el acoplamiento entre componentes y facilita tanto el mantenimiento como la evolución del sistema. La Ilustración 5 resume dicha arquitectura.

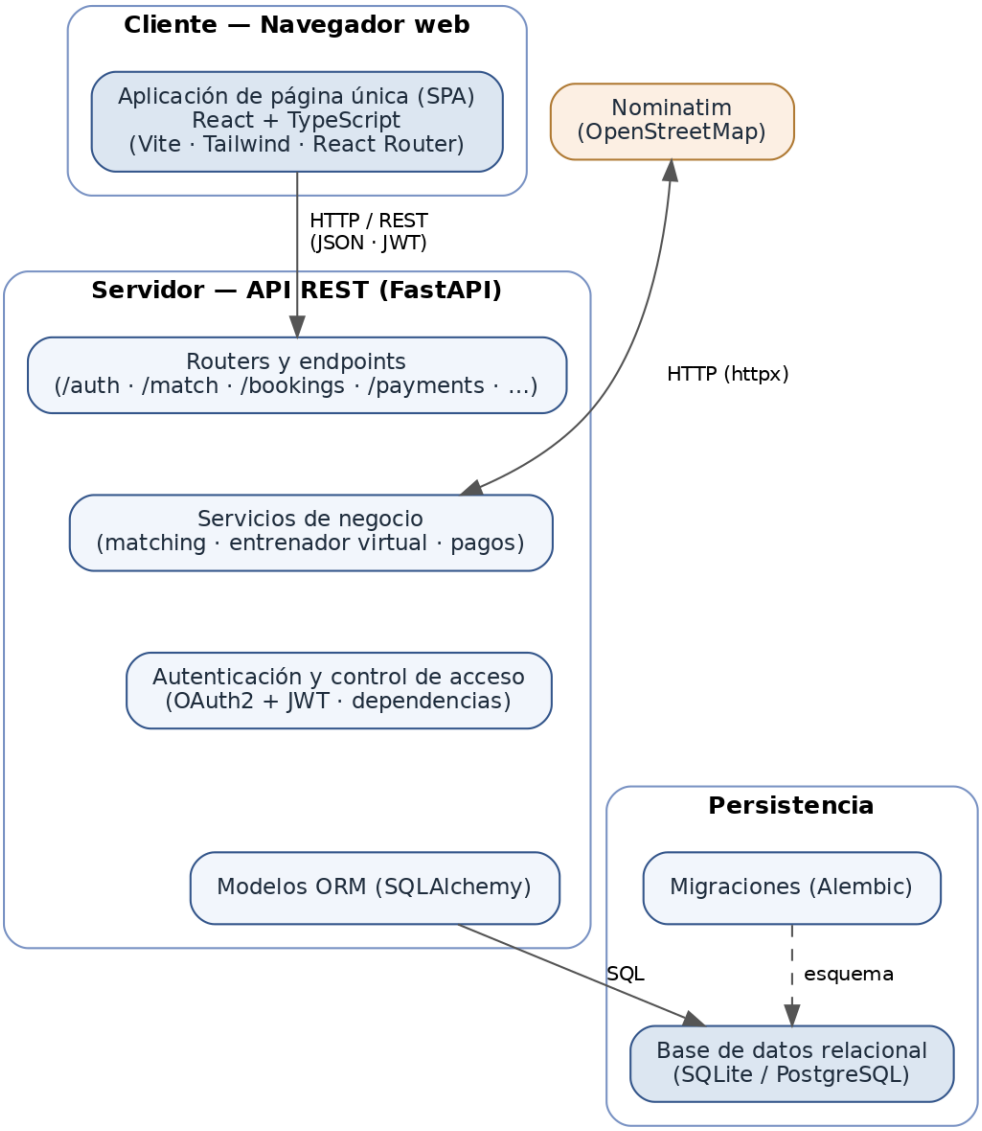

Ilustración 5: Arquitectura general de PlayMatch.

En la capa de presentación, la aplicación de página única se ejecuta en el navegador y se comunica con el servidor exclusivamente a través de la API REST, intercambiando mensajes en formato JSON. La autenticación es sin estado: el cliente adjunta en cada petición un token JWT que el servidor verifica, de modo que la API no necesita mantener sesiones en memoria. La capa de servidor, construida con FastAPI, se organiza internamente en varios niveles. Los routers y endpoints reciben las peticiones y validan los datos de entrada con Pydantic; delegan la lógica en los servicios de negocio, como el matching, el entrenador virtual o el cálculo de los pagos; y las dependencias de seguridad resuelven la autenticación

y el control de acceso antes de ejecutar cada operación. El acceso a los datos se realiza a través de los modelos ORM de SQLAlchemy, que traducen las operaciones a SQL sobre la base de datos relacional. Por último, la geocodificación opcional de direcciones se apoya en el servicio externo Nominatim, al que la capa de servicios accede mediante peticiones HTTP.

5.2.2 MODELO DE DOMINIO Y ROLES

El dominio de PlayMatch gira en torno a dos conceptos: el usuario y la reserva de clases. Cada usuario tiene un rol —alumno, profesor o administrador— que determina las operaciones que puede realizar, y dispone de un perfil asociado en una relación uno a uno. El perfil reutiliza el mismo campo para el precio por hora: en un profesor representa su tarifa y en un alumno, su presupuesto máximo, lo que simplifica el modelo sin perder expresividad. Alrededor de estas dos entidades se articulan la disponibilidad publicada por los profesores, las reservas, los pagos, las valoraciones, los mensajes de chat y las notificaciones. La *Tabla 5* resume las entidades principales y sus relaciones.

Tabla 5: Entidades principales del modelo de dominio.

Entidad	Descripción	Relaciones clave
User	Cuenta de usuario con sus credenciales y su rol (alumno, profesor o administrador).	1:1 con Profile; origen de la mayoría de relaciones.
Profile	Datos del usuario relevantes para el matching: ubicación, nivel y precio o presupuesto.	1:1 con User.
AvailabilitySlot	Franja horaria en la que un profesor está disponible para impartir clase.	N:1 con User (profesor).
Booking	Reserva de una clase, con su estado y los datos de cancelación e incidencia.	N:1 con User (alumno y profesor) y con AvailabilitySlot.
Payment	Pago asociado a una reserva, con su estado de escrow e importes.	N:1 con Booking.
PaymentMethod	Método de pago (tarjeta) guardado por un usuario, con datos parciales.	N:1 con User.
ChatMessage	Mensaje del chat asociado a una reserva.	N:1 con Booking y con User (emisor).
ChatReadState	Marca de última lectura del chat por usuario y reserva.	N:1 con Booking y con User.
Review	Valoración multidimensional de una clase completada.	1:1 con Booking; N:1 con User (autor y valorado).
Notification	Aviso dirigido a un usuario sobre un evento del sistema.	N:1 con User; N:1 opcional con Booking.

5.2.3 ESQUEMA RELACIONAL

El modelo de datos se ha implementado sobre una base de datos relacional mediante el ORM SQLAlchemy, y su evolución se gestiona con migraciones de Alembic. La Ilustración 6 muestra el esquema resultante, con las entidades, sus campos principales y las relaciones establecidas a través de claves foráneas.

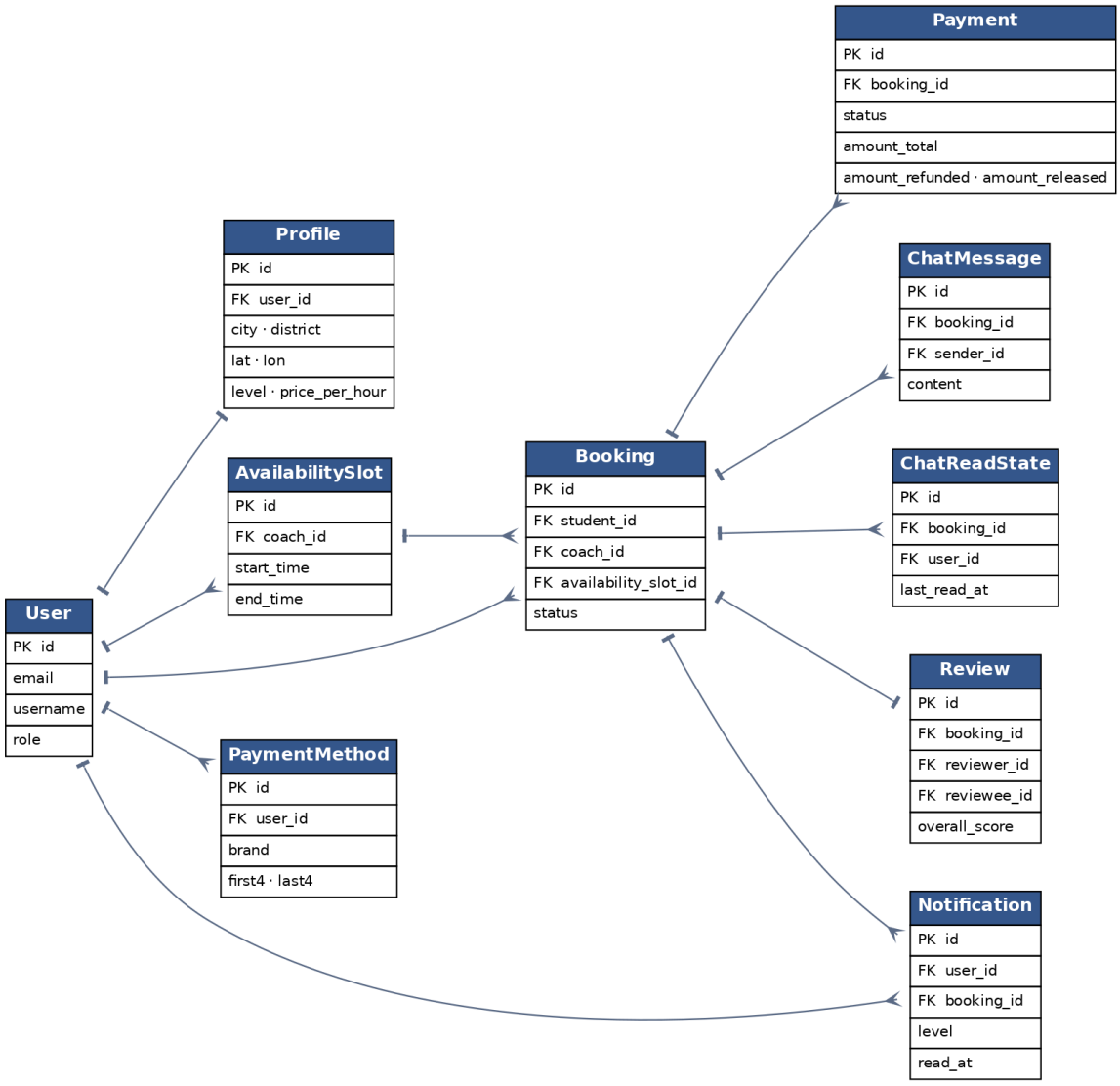

Ilustración 6: Modelo de datos relacional de PlayMatch.

El usuario actúa como entidad central del esquema. De él dependen el perfil, las franjas de disponibilidad —cuando el usuario es profesor— y los métodos de pago. La reserva es la otra entidad nuclear: enlaza al alumno, al profesor y a la franja reservada, y a partir de ella cuelgan el pago, la valoración, los mensajes del chat, los estados de lectura y las notificaciones. Algunas relaciones son de uno a uno, como la que une al usuario con su perfil o a la reserva con su valoración; el resto son de uno a varios. Para no recargar el diagrama, no se han representado todas las claves foráneas que apuntan al usuario: además de las mostradas, el mensaje de chat guarda a su emisor, la valoración distingue al autor y al profesor valorado, y el estado de lectura se asocia a cada usuario.

5.2.4 CICLO DE VIDA DE LA RESERVA

La reserva es la entidad con un ciclo de vida más rico, y su gestión correcta resulta clave para la fiabilidad de la plataforma. La Ilustración 7 representa los estados por los que puede pasar una reserva y las transiciones que los conectan.

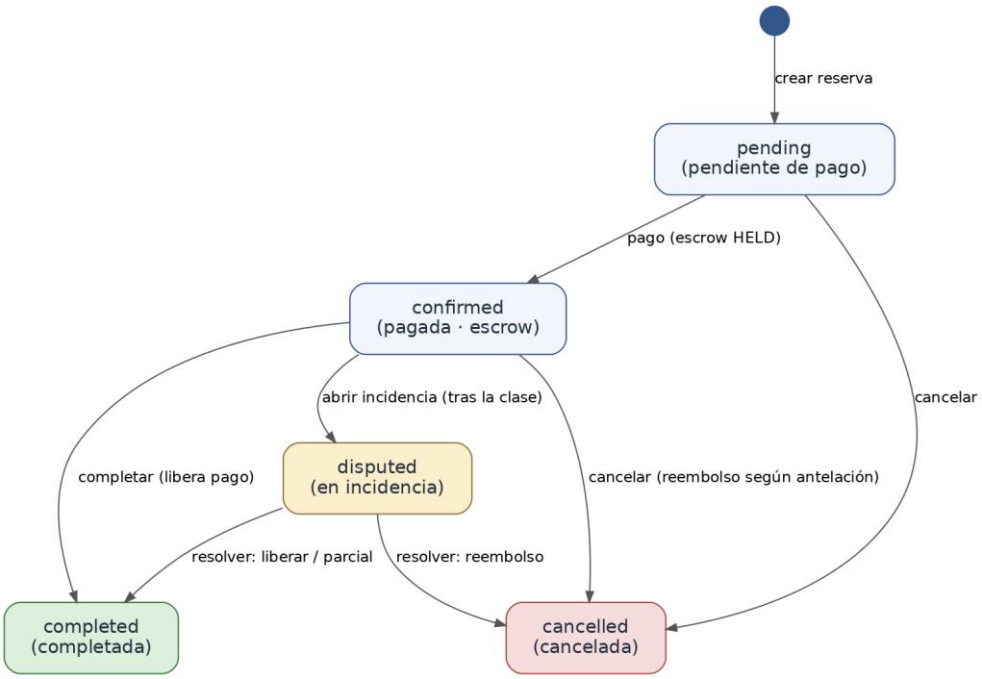

Ilustración 7: Diagrama de estados de la reserva.

Tabla 6: Estados de la reserva.

Estado	Descripción
pending	Reserva creada y pendiente de pago; la franja queda bloqueada para otros alumnos.
confirmed	El alumno ha pagado y el importe queda retenido en escrow a la espera de que se imparta la clase.
completed	La clase se ha impartido; el pago retenido se libera al profesor.
cancelled	La reserva se ha cancelado; el importe se reembolsa total o parcialmente según quién cancele y la antelación.
disputed	El alumno ha abierto una incidencia tras la clase; el importe permanece retenido hasta que el administrador la resuelve.

El estado de la reserva va acompañado del estado del pago, que evoluciona en paralelo a través de los valores de retención (HELD), liberación (RELEASED), reembolso (REFUNDED) o reparto parcial (PARTIAL). Las reglas concretas que rigen estas transiciones —plazos de cancelación, porcentajes de reembolso y resolución de incidencias— se detallan en el capítulo 5.5.

5.2.5 FLUJO DE RESERVA, PAGO Y NOTIFICACIÓN

Para ilustrar cómo cooperan las distintas capas, la Ilustración 8 describe mediante un diagrama de secuencia el flujo más representativo del sistema: desde que el alumno reserva una clase hasta que el pago queda retenido y se generan las notificaciones correspondientes.

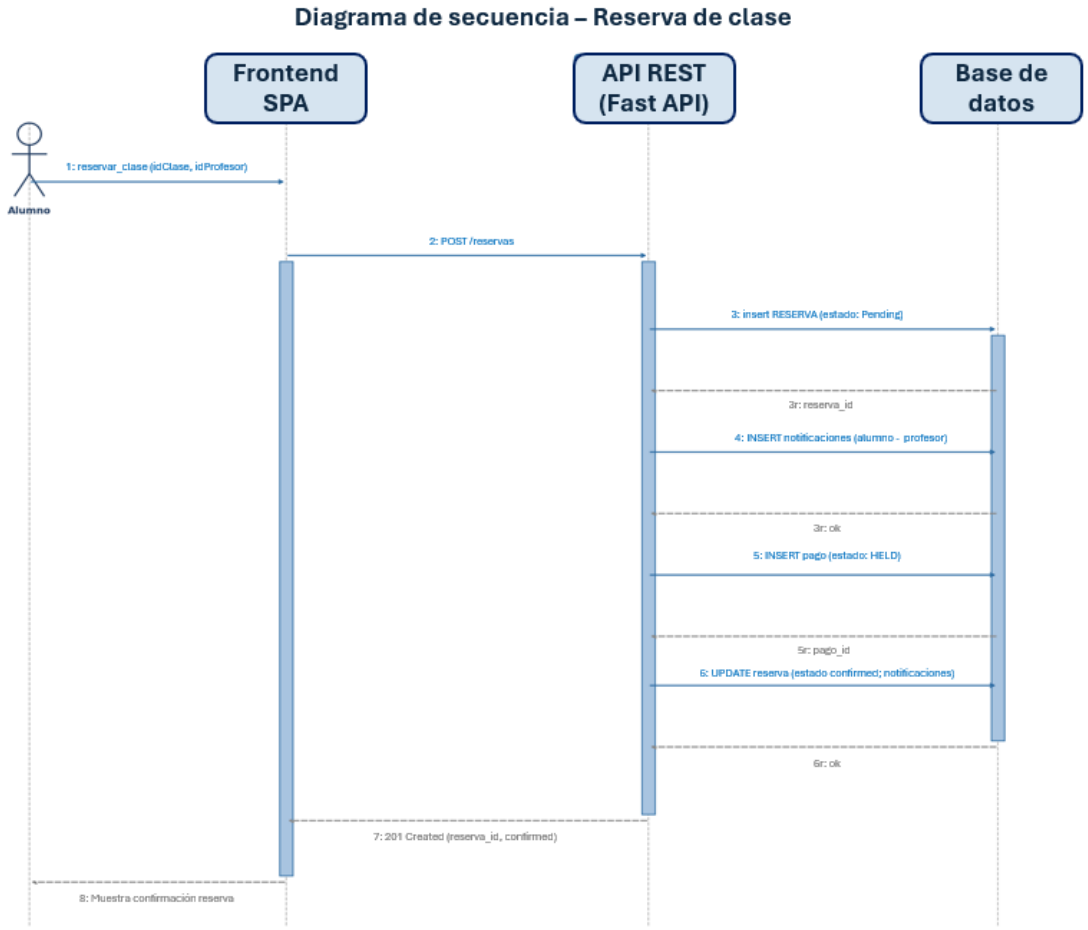

Ilustración 8: Secuencia de reserva, pago y notificación.

El alumno inicia el proceso desde la interfaz, que traduce sus acciones en llamadas a la API. La creación de la reserva la deja en estado pendiente de pago y notifica a ambas partes. Cuando el alumno paga, la API valida los datos de la tarjeta, registra el pago en estado retenido, confirma la reserva y emite nuevas notificaciones. En todo momento la lógica de negocio reside en el servidor, mientras que la interfaz se limita a orquestar las peticiones y a mostrar el resultado al usuario.

5.2.6 SÍNTESIS

El diseño descrito en este capítulo ofrece una visión estática del sistema, a través de la arquitectura y el modelo de datos, y una visión dinámica, mediante los estados de la reserva y los flujos entre componentes. Sobre esta base, el capítulo siguiente profundiza en

el componente que da nombre a la plataforma y constituye su principal aportación técnica: el algoritmo de matching.

5.3 ALGORITMO DE MATCHING Y UBICACIÓN

El algoritmo de matching constituye el componente central de PlayMatch y su principal aportación técnica. Su cometido es proponer a cada alumno, entre todos los profesores disponibles, aquellos cuya clase se ajusta mejor a su perfil y a sus preferencias. No se trata de un sistema de aprendizaje automático, sino de un procedimiento determinista en dos fases: un filtrado previo que descarta a los candidatos no válidos y una función de puntuación que ordena a los que superan ese filtro. Este enfoque prioriza la transparencia y la reproducibilidad, ya que, ante las mismas entradas, el sistema devuelve siempre el mismo resultado y permite explicar por qué un profesor aparece antes que otro.

5.3.1 OBJETIVO Y DATOS DE ENTRADA

El matching recibe dos entradas: el perfil del alumno autenticado y la ventana horaria en la que desea recibir la clase. Del perfil se utilizan los atributos que permiten valorar la afinidad con cada profesor, recogidos en la *Tabla 7*. La ciudad actúa además como condición de entrada: si el alumno no tiene una ciudad válida o carece de coordenadas, el sistema no devuelve resultados, ya que no podría calcular distancias con sentido.

Tabla 7: Atributos del perfil del alumno utilizados en el matching.

Atributo	**Uso en el matching**
Ciudad	Debe ser una ciudad admitida; restringe la búsqueda a profesores de la misma ciudad.
Distrito	Determina, al guardar el perfil, las coordenadas aproximadas del alumno.
Latitud y longitud	Permiten calcular la distancia con cada profesor; son obligatorias para el cálculo.
Nivel (1–5)	Penaliza la diferencia de nivel entre el alumno y el profesor.
Precio por hora	Como presupuesto máximo del alumno; penaliza a los profesores cuya tarifa lo supera.

5.3.2 GESTIÓN DE LA UBICACIÓN: CIUDADES Y DISTRITOS

La ubicación se gestiona mediante un catálogo propio de ciudades y distritos. Cada distrito lleva asociadas unas coordenadas aproximadas correspondientes a su centro, de modo que, al guardar su perfil, el usuario solo elige ciudad y distrito y el sistema deriva automáticamente la latitud y la longitud. La *Tabla 8* recoge las ciudades admitidas y el número de distritos de cada una.

Tabla 8: Ciudades admitidas y número de distritos.

Ciudad	Distritos	Ejemplos
Madrid	21	Centro, Chamberí, Salamanca, Hortaleza…
Barcelona	10	Ciutat Vella, Eixample, Gràcia…
Valencia	19	Ciutat Vella, L'Eixample, Benimaclet…
Sevilla	11	Casco Antiguo, Triana, Nervión…

El nombre de la ciudad se normaliza a una forma canónica para evitar inconsistencias, y el matching nunca cruza ciudades: un alumno de Madrid no verá profesores de Barcelona aunque coincidan en horario. De forma complementaria, la aplicación permite geocodificar texto libre a través de Nominatim, si bien el flujo habitual se apoya en el catálogo de distritos.

5.3.3 FILTRADO DE CANDIDATOS

Antes de calcular ninguna puntuación, el sistema reduce el conjunto de profesores a los candidatos viables aplicando, en orden, una serie de filtros. La Ilustración 9 muestra este proceso.

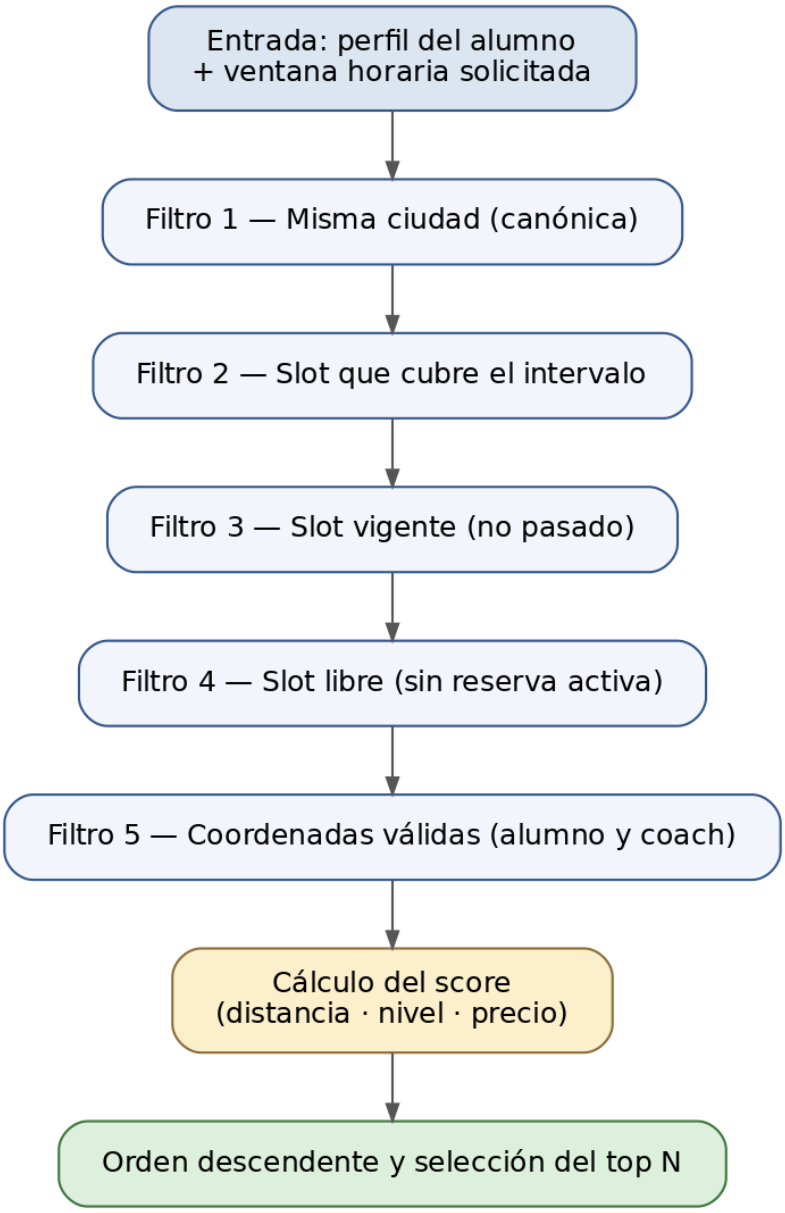

Ilustración 9: Proceso de filtrado y puntuación del matching.

El primer filtro, y el más restrictivo, exige que el profesor pertenezca a la misma ciudad que el alumno. El segundo selecciona únicamente las franjas de disponibilidad que cubren por completo el intervalo solicitado. El tercero descarta las franjas ya pasadas. El cuarto elimina las que tienen una reserva activa, es decir, en cualquier estado distinto de cancelada. El quinto y último exige que tanto el alumno como el profesor dispongan de coordenadas válidas. Solo los pares de profesor y franja que superan los cinco filtros pasan a la fase de puntuación.

5.3.4 FUNCIÓN DE PUNTUACIÓN

A cada candidato se le asigna una puntuación, o score, que mide su grado de ajuste al alumno. Esta puntuación se construye en dos pasos: primero se calcula un coste que combina tres factores y, después, ese coste se transforma en una puntuación acotada entre cero y uno. La Ilustración 10 recoge las expresiones empleadas.

$$d = 2r \arcsin\sqrt{\sin^2(\Delta\varphi/2) + \cos\varphi_1 \cos\varphi_2 \sin^2(\Delta\lambda/2)} \quad (r = 6371 \text{ km})$$

$$\text{coste} = 0{,}5\, d \;+\; 0{,}3\, \Delta\text{nivel} \;+\; 0{,}2\, \max(0,\; \text{precio} - \text{presupuesto})$$

$$\text{score} = \frac{1}{1 + \text{coste}}$$

Ilustración 10: Cálculo de la distancia, el coste y la puntuación del matching.

El primer factor es la distancia geográfica entre alumno y profesor, calculada con la fórmula de Haversine a partir de sus coordenadas. El segundo es la diferencia de nivel, en valor absoluto. El tercero es una penalización de precio, que solo actúa cuando la tarifa del profesor supera el presupuesto del alumno. Los tres factores se combinan en un coste mediante una suma ponderada, y la puntuación final es la inversa de uno más el coste, de manera que un coste menor produce una puntuación mayor. La *Tabla 9*resume los factores y los pesos aplicados.

Tabla 9: Factores de la función de puntuación y sus pesos.

Factor	Cálculo	Peso
Distancia	Distancia de Haversine entre alumno y profesor, en kilómetros	0,5
Nivel	Valor absoluto de la diferencia de nivel	0,3
Precio	max(0, tarifa del profesor − presupuesto del alumno)	0,2

Los pesos otorgan más importancia a la cercanía que al nivel y al precio, en coherencia con la naturaleza presencial de las clases: para el alumno suele ser prioritario que el profesor esté cerca. Son unos pesos defendibles y fáciles de interpretar, fijados de forma razonada; su ajuste fino, e incluso su personalización por usuario, queda planteado como línea de mejora. Conviene notar que ni la ciudad ni el distrito puntúan por separado: la ciudad ya ha actuado como filtro y el distrito influye únicamente a través de las coordenadas que determinan la distancia.

5.3.5 EJEMPLO NUMÉRICO

Para ilustrar el cálculo, considérese un alumno situado en el distrito de Chamberí, con nivel 3 y un presupuesto de 30 euros por hora. Tras los filtros quedan dos profesores candidatos en Madrid: el profesor A, en el distrito Centro, con nivel 3 y una tarifa de 28 euros; y el profesor B, en Hortaleza, con nivel 4 y una tarifa de 35 euros. La *Tabla 10* muestra el cálculo para cada uno.

Tabla 10: Ejemplo de cálculo de la puntuación para dos candidatos.

Candidato	Distancia (km)	Δ nivel	Penal. precio (€)	Coste	Score
Profesor A (Centro)	1,93	0	0	0,966	0,509
Profesor B (Hortaleza)	6,74	1	5	4,671	0,176

El profesor A obtiene una puntuación claramente superior: está más cerca, comparte el nivel del alumno y su tarifa no supera el presupuesto, por lo que su coste es bajo. El profesor B, más alejado, con un nivel por encima y una tarifa que excede el presupuesto en cinco euros, acumula un coste mucho mayor y queda relegado. En consecuencia, el sistema mostraría en primer lugar al profesor A.

5.3.6 INTEGRACIÓN EN LA BÚSQUEDA

En la interfaz, la búsqueda no se limita a una única consulta. El alumno elige un día y una franja —toda la jornada, la mañana, la tarde o una hora concreta— y la aplicación realiza una consulta al algoritmo por cada hora candidata, empleando ventanas de una hora. A continuación, fusiona los resultados por franja de disponibilidad, conservando para cada una la mejor puntuación obtenida, y presenta el conjunto ordenado de mejores opciones. Desde los resultados, el alumno puede reservar directamente la clase elegida.

5.3.7 SÍNTESIS Y LIMITACIONES

El algoritmo descrito resuelve el problema de emparejamiento con un coste computacional reducido y un comportamiento explicable, lo que encaja con el alcance de un producto mínimo viable. Sus principales limitaciones marcan, a la vez, su hoja de ruta: el distrito podría incorporarse a la puntuación con un peso propio, la distancia podría normalizarse para acotar su efecto y la reputación del profesor, a partir de sus valoraciones, podría integrarse como un factor más. Estas mejoras se retoman en el capítulo de conclusiones.

5.4 SEGURIDAD Y PROTECCIÓN DE DATOS SENSIBLES

PlayMatch trata datos personales y, en algunos casos, sensibles, por lo que la seguridad ha sido una consideración presente desde las primeras decisiones de diseño. Este capítulo describe las medidas adoptadas para proteger la información: comienza identificando los datos sensibles que maneja la plataforma y continúa con la autenticación, la autorización, la seguridad de los pagos, la protección de las comunicaciones y el marco de protección de datos personales. Se cierra con un modelo de amenazas simplificado y un balance entre lo implementado y lo que queda pendiente. En el ámbito normativo, la plataforma se enmarca en el Reglamento General de Protección de Datos (RGPD, Reglamento UE 2016/679) y en su transposición al ordenamiento español mediante la Ley Orgánica 3/2018, de Protección de Datos Personales y garantía de los derechos digitales (LOPDGDD).

5.4.1 DATOS SENSIBLES TRATADOS

El primer paso para proteger la información es saber qué información se maneja, dónde se almacena y quién puede acceder a ella. La *Tabla 11* recoge los principales datos sensibles de PlayMatch junto con su nivel de sensibilidad.

Tabla 11: Datos sensibles tratados por PlayMatch.

Dato	Dónde se almacena	Quién accede	Sensibilidad
Contraseñas	Tabla de usuarios, cifradas con bcrypt	Nadie en claro; el sistema solo las verifica	Muy alta
Identidad (correo, usuario, perfil)	Tablas de usuarios y perfiles	El propio usuario; datos públicos limitados	Media

Dato	Dónde se almacena	Quién accede	Sensibilidad
Ubicación (ciudad, distrito, coordenadas)	Tabla de perfiles	El propio usuario; la usa el matching	Media
Mensajes de chat	Tabla de mensajes	Solo los participantes de la reserva	Media-alta
Medios de pago (datos parciales)	Tabla de métodos de pago	El propio usuario	Alta
Historial financiero (pagos, escrow)	Tabla de pagos	Participantes de la reserva y administrador	Alta

5.4.2 AUTENTICACIÓN

La autenticación verifica la identidad de quien usa la plataforma. En el registro, la contraseña no se guarda nunca en claro: se transforma con el algoritmo bcrypt, una función de hash diseñada específicamente para contraseñas, y solo se almacena el resultado. En el inicio de sesión, el sistema vuelve a aplicar bcrypt sobre la contraseña introducida y la compara con el valor guardado. Si coincide, emite un token JWT firmado con una clave secreta, que incluye el identificador del usuario, su rol y una fecha de caducidad. El intercambio sigue el flujo OAuth2 de tipo password, un estándar ampliamente adoptado. A partir de ese momento, el cliente conserva el token y lo adjunta en cada petición; como el token contiene la información necesaria y va firmado, la autenticación es sin estado y el servidor no necesita mantener sesiones. La clave de firma se gestiona como variable de entorno y debe sustituirse por un valor robusto en producción.

5.4.3 AUTORIZACIÓN Y CONTROL DE ACCESO

Una vez identificado el usuario, el control de acceso decide qué puede hacer. La Ilustración 11 resume el flujo completo, desde el inicio de sesión hasta la autorización de cada operación.

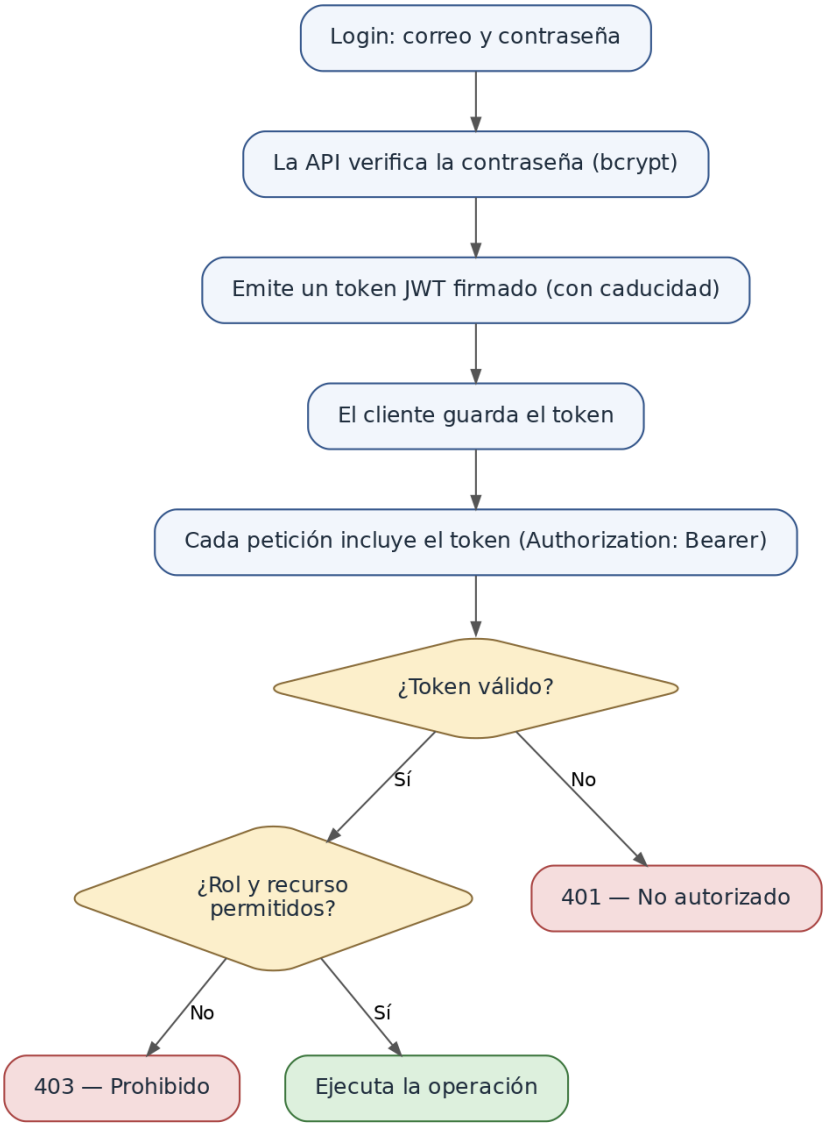

Ilustración 11: Flujo de autenticación y control de acceso.

El control opera en dos niveles. El primero es la autorización por rol: una dependencia común a los endpoints protegidos decodifica el token, recupera al usuario y comprueba que su rol le permite la operación; si el token falta o no es válido, la API responde con un error 401, y si el rol no autoriza la acción, con un 403. Así, por ejemplo, solo un alumno puede reservar o solicitar recomendaciones, solo un profesor puede publicar disponibilidad y solo un administrador puede resolver incidencias. El segundo nivel es la autorización por recurso: en las operaciones ligadas a una reserva concreta, el sistema verifica además que el usuario sea uno de sus participantes. De este modo, un alumno no puede acceder al chat ni al pago de la reserva de otro, aunque su rol sea el adecuado.

Únicamente quedan abiertos al público el registro, el inicio de sesión y la comprobación de estado del servidor; el resto de la API exige un token válido. La *Tabla 12* resume qué operaciones puede realizar cada rol.

Tabla 12: Matriz de acceso por rol.

Operación	**Alumno**	**Profesor**	**Administrador**
Registro e inicio de sesión	Sí	Sí	Sí
Gestión del perfil propio	Sí	Sí	Sí
Publicación de disponibilidad	No	Sí	No
Búsqueda de clases (matching)	Sí	No	No
Reserva, pago y finalización	Sí	No	No
Valoración del profesor	Sí	No	No
Apertura de incidencia	Sí	No	No
Respuesta a una incidencia	No	Sí	No
Resolución de una incidencia	No	No	Sí
Cancelación de una reserva	Sí	Sí	Sí
Chat de una reserva	Sí	Sí	No

Las operaciones ligadas a una reserva concreta —el chat, el pago o la cancelación— están además restringidas a los usuarios que participan en ella, con independencia de su rol.

5.4.4 SEGURIDAD EN LOS PAGOS Y EL MONEDERO

El tratamiento de los medios de pago es uno de los puntos más delicados. En PlayMatch la pasarela es una simulación, por lo que no se procesan cobros reales ni se persigue el cumplimiento del estándar PCI-DSS propio de los pagos auténticos. Aun así, el diseño aplica el principio de minimización: de cada tarjeta guardada en el monedero solo se conservan datos parciales —los cuatro primeros y los cuatro últimos dígitos, la marca, el titular y la caducidad—, y nunca el número completo ni el código de seguridad. Los datos introducidos se validan en el servidor (longitud y formato del número, caducidad y código), pero el código de seguridad se emplea solo para esa validación y no se almacena. Cada pago queda asociado a una referencia que facilita su trazabilidad. El principal riesgo residual es, precisamente, la ausencia de una pasarela real; su sustitución por una solución tokenizada, en la que los datos de la tarjeta nunca lleguen al servidor, es la principal línea de trabajo futuro en este apartado.

5.4.5 SEGURIDAD DE LAS COMUNICACIONES Y DE LA API

En el plano de las comunicaciones, la API restringe mediante CORS los orígenes desde los que se aceptan peticiones, limitándolos a los del entorno de desarrollo y de la demostración. Toda la información que entra en el sistema se valida con Pydantic antes de procesarse, lo que reduce la superficie de ataque frente a entradas mal formadas. Conviene señalar un aspecto honesto de la versión actual: ante un error inesperado, el manejador devuelve, junto al código 500, el mensaje de la excepción, lo que resulta cómodo durante el desarrollo pero debería ocultarse en producción para no revelar detalles internos. Del mismo modo, en un despliegue real las comunicaciones deberían viajar siempre sobre HTTPS.

5.4.6 PROTECCIÓN DE DATOS PERSONALES (MARCO RGPD Y LEGISLACIÓN ESPAÑOLA)

Más allá de las medidas técnicas, conviene encuadrar el tratamiento de datos personales en el marco del Reglamento General de Protección de Datos, aunque sea de forma orientativa dado el carácter académico del proyecto. La finalidad del tratamiento es acotada y explícita: gestionar la relación entre alumnos y profesores dentro de la plataforma. La base legitimadora más natural sería la ejecución del contrato de prestación del servicio. El diseño aplica de forma efectiva dos principios del reglamento: la minimización de los datos, especialmente visible en los medios de pago, y la limitación del acceso por rol y por recurso. En cuanto a los derechos de los usuarios, el acceso y la rectificación se cubren a través de la edición del perfil, mientras que la supresión y la portabilidad automatizadas quedan como trabajo futuro. Como limitación propia de un MVP, el sistema no implementa la exportación ni el borrado automatizado de datos, ni formaliza un acuerdo de tratamiento con el servicio externo de geocodificación.

5.4.7 BUENAS PRÁCTICAS DE DESPLIEGUE

La seguridad no termina en el código. Entre las buenas prácticas adoptadas y recomendadas para el despliegue figuran mantener la configuración sensible —como la clave de firma de los tokens— en variables de entorno y fuera del control de versiones, rotar dicha clave en producción, no versionar la base de datos, servir siempre la aplicación sobre HTTPS y separar con claridad los entornos de desarrollo y de producción.

5.4.8 MODELO DE AMENAZAS SIMPLIFICADO

Para sintetizar la postura de seguridad, la *Tabla 13* relaciona los principales activos del sistema con las amenazas que podrían afectarlos y con las medidas que las mitigan.

Tabla 13: Modelo de amenazas simplificado.

Activo	Amenaza	Mitigación
Contraseñas	Robo o filtración de credenciales	Cifrado con bcrypt; nunca se almacenan ni transmiten en claro.
Token de sesión	Suplantación de identidad	JWT firmado con clave secreta y con fecha de caducidad.
Datos de una reserva	Acceso por usuarios no implicados	Control de acceso por recurso: solo los participantes.
Medios de pago	Exposición de datos de tarjeta	Minimización: sin número completo ni código de seguridad.
API	Peticiones desde orígenes no autorizados	Restricción de CORS a los orígenes conocidos.
Datos personales	Tratamiento indebido	Minimización y limitación de acceso; orientación al RGPD.

5.4.9 RESUMEN: MEDIDAS IMPLEMENTADAS Y PENDIENTES

Por último, la *Tabla 14*ofrece un balance entre las medidas de seguridad ya implementadas y las que quedan pendientes o solo abordadas a nivel conceptual. Este balance, lejos de ocultar las carencias del MVP, demuestra criterio al distinguir lo que un producto académico puede garantizar de lo que exigiría un producto en explotación.

Tabla 14: Medidas de seguridad implementadas y pendientes.

Medida	Estado
Cifrado de contraseñas (bcrypt)	Implementado
Autenticación con JWT y caducidad	Implementado
Autorización por rol y por recurso	Implementado
Minimización de datos de pago (sin PAN ni CVV)	Implementado
Restricción de CORS	Implementado
Validación de entradas con Pydantic	Implementado
Pasarela de pago real / PCI-DSS	Pendiente (pago simulado)
HTTPS y secretos robustos en producción	Pendiente (documentado)
RGPD operativo (exportación y borrado)	Parcial (marco teórico)

Medida	**Estado**
Pruebas de seguridad automatizadas	Parcial (pruebas de control de acceso)

5.5 REGLAS DE NEGOCIO Y ENTRENADOR VIRTUAL

Más allá de su arquitectura, PlayMatch incorpora una serie de reglas de negocio que gobiernan el dinero, las cancelaciones y las incidencias, y que resultan determinantes para generar confianza entre las partes. Este capítulo describe esas reglas —el esquema de pago en retención, la política de cancelación, la gestión de incidencias y las valoraciones— y termina con el entrenador virtual, una funcionalidad de apoyo al alumno con un enfoque deliberadamente sencillo.

5.5.1 EL ESQUEMA DE PAGO EN RETENCIÓN (ESCROW)

El pago en PlayMatch no llega de inmediato al profesor, sino que sigue un esquema de retención, o escrow, inspirado en las plataformas de intermediación. Cuando el alumno paga, el importe queda retenido y solo se libera al profesor una vez impartida la clase; si la reserva se cancela o una incidencia se resuelve a favor del alumno, el importe se le devuelve total o parcialmente. Para reflejar esta lógica, cada pago registra el importe total y las cantidades reembolsadas y liberadas, y evoluciona a través de los estados que recoge la *Tabla 15*.

Tabla 15: Estados del pago en el esquema de escrow.

Estado	Significado
HELD (retenido)	El pago se ha realizado y el importe queda retenido a la espera de la clase.
RELEASED (liberado)	La clase se ha completado y el importe se ha liberado al profesor.
REFUNDED (reembolsado)	El importe se ha devuelto íntegramente al alumno.
PARTIAL (parcial)	Una parte se ha devuelto al alumno y otra se ha liberado al profesor.
FAILED (fallido)	El intento de pago no se completó con éxito.

5.5.2 CANCELACIÓN Y REEMBOLSOS

Las reservas pueden cancelarse, pero las consecuencias económicas dependen de quién cancela y de la antelación con la que lo hace. La política, resumida en la *Tabla 16*, busca un equilibrio entre la flexibilidad para el alumno y la protección del profesor frente a las cancelaciones de última hora.

Tabla 16: Política de cancelación y reembolsos.

Quién cancela	Antelación	Reembolso al alumno	Importe al profesor	Penalización
Alumno	5 días o más	100 %	0 %	No
Alumno	Menos de 5 días	25 %	75 %	No
Profesor	Cualquiera	100 %	0 %	Sí

Cuando el alumno cancela con menos de cinco días, el profesor recibe el 75 % del importe como compensación; cuando es el profesor quien cancela, el alumno recupera todo su dinero y se anota una penalización en la reserva. Además, tras una cancelación tardía del alumno, el profesor puede volver a publicar la franja liberada desde sus notificaciones, de modo que el hueco quede de nuevo disponible para otros alumnos.

5.5.3 INCIDENCIAS Y DISPUTAS

No todas las clases se desarrollan sin problemas. Si tras la clase el alumno considera que algo no fue como debía, puede abrir una incidencia, siempre que el pago siga retenido. La reserva pasa entonces a un estado de disputa y el profesor dispone de un plazo de tres días para enviar sus explicaciones. La resolución corresponde al administrador, que actúa como árbitro y cuenta con tres acciones posibles, recogidas en la *Tabla 17*.

Tabla 17: Acciones de resolución de una incidencia.

Acción del administrador	Efecto sobre el pago	Estado final de la reserva
Reembolso	Devuelve el importe íntegro al alumno	cancelled
Liberación	Libera el importe íntegro al profesor	completed
Reparto	Reparte el importe según un porcentaje	completed

Este procedimiento garantiza que ningún importe se libere de forma automática mientras exista una controversia y que la decisión final recaiga en una figura neutral.

5.5.4 VALORACIONES MULTIDIMENSIONALES

Una vez completada la clase, el alumno puede valorar al profesor. A diferencia de una valoración de una sola estrella, PlayMatch emplea una valoración multidimensional: el alumno puntúa nueve aspectos distintos de la clase en una escala de uno a cinco y puede

añadir un comentario. A partir de esas nueve puntuaciones, el sistema calcula una valoración global que se muestra, junto con las reseñas, en el perfil público del profesor. La *Tabla 18* recoge los nueve ejes valorados.

Tabla 18: Ejes de la valoración del profesor.

Eje	**Qué valora**
Dinamismo	El ritmo y la energía de la clase.
Diversión	Lo entretenida que resulta la sesión.
Variedad	La diversidad de ejercicios y situaciones.
Claridad	La claridad de las explicaciones.
Corrección técnica	La calidad de las correcciones del profesor.
Adaptación al nivel	El ajuste de la clase al nivel del alumno.
Puntualidad	El cumplimiento de los horarios.
Organización	La preparación y la estructura de la sesión.
Actitud	El trato y la actitud del profesor.

5.5.5 ENTRENADOR VIRTUAL

PlayMatch incorpora un entrenador virtual que ofrece al alumno orientación técnica básica. Su diseño es deliberadamente sencillo: se trata de un asistente determinista basado en reglas y palabras clave, que no recurre a modelos de lenguaje ni a servicios externos de inteligencia artificial. Esta decisión, ya justificada en capítulos anteriores, prioriza la reproducibilidad, el coste nulo de operación y la explicabilidad de las respuestas. El asistente está reservado a los alumnos e incluye en cada respuesta un aviso de que sus consejos son orientativos y no sustituyen al profesor. La Ilustración 12 resume su funcionamiento.

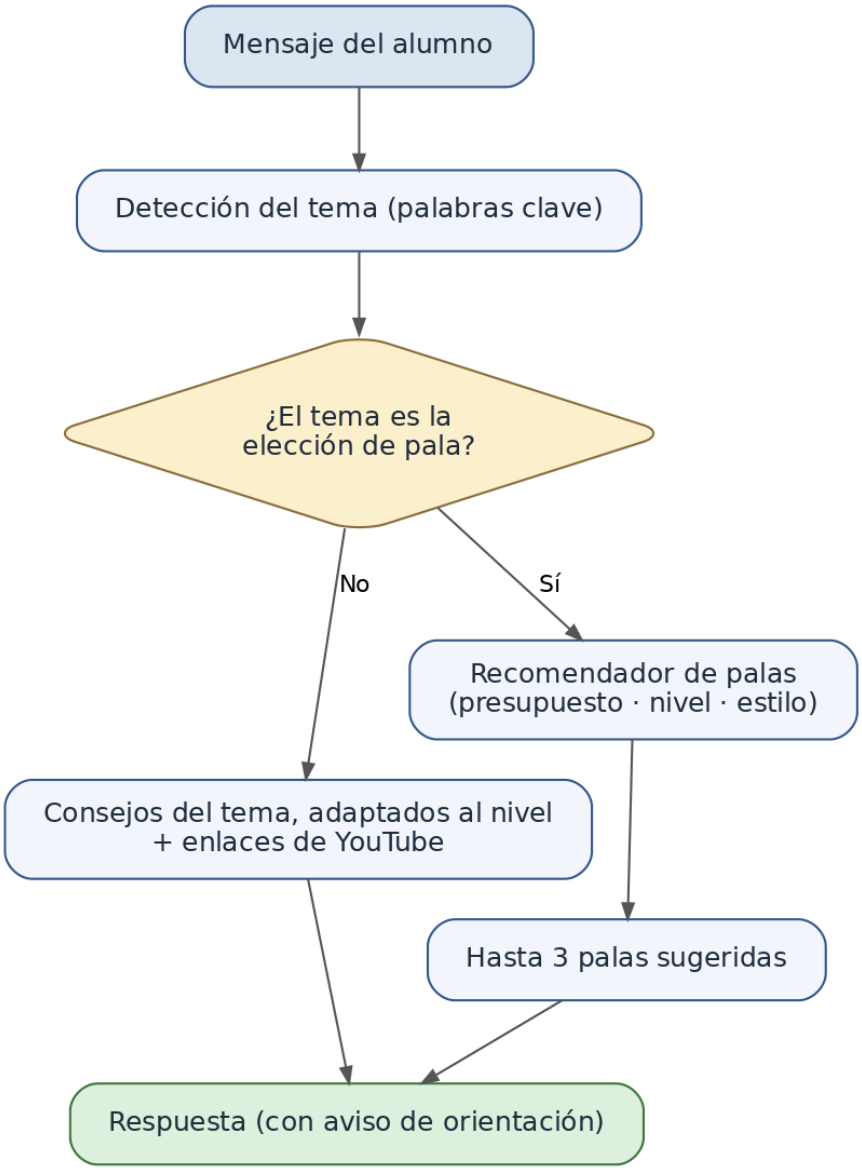

Ilustración 12: Flujo de funcionamiento del entrenador virtual.

5.5.6 FASE 1: CONSEJOS TÉCNICOS Y VÍDEOS

En su primera fase, el entrenador detecta el tema de la consulta a partir de palabras clave y responde con una serie de consejos numerados, adaptados al nivel del alumno, acompañados de hasta dos enlaces de búsqueda en YouTube sobre ese tema. La *Tabla 19* muestra los temas que reconoce.

Tabla 19: Temas reconocidos por el entrenador virtual.

Tema	Palabras clave de ejemplo
Bandeja y revés	bandeja, revés, drive
Volea	volea, bloqueo, pared
Remate	remate, smash, definir
Saque y resto	saque, servicio, resto
Posición en pista	posición, desplazamiento, split
Elección de pala	pala, comprar, presupuesto

5.5.7 FASE 2: RECOMENDACIÓN DE PALAS

Cuando la consulta trata sobre la elección de pala, el asistente activa su segunda fase. A partir de un catálogo estático de doce modelos de referencia —con su precio orientativo, forma, peso, estilo y rango de nivel—, el sistema interpreta del mensaje el presupuesto, el nivel y el estilo de juego deseados, puntúa los modelos compatibles y devuelve hasta tres recomendaciones con su justificación. En la interfaz, este proceso puede guiarse mediante un breve cuestionario de tres pasos (tipo de juego, nivel y presupuesto) o resolverse en un único mensaje que incluya toda la información. El catálogo es estático y los precios son orientativos: no se consultan tiendas ni precios en tiempo real, lo que de nuevo prioriza la reproducibilidad de la demostración.

5.5.8 SÍNTESIS

Las reglas de negocio descritas dotan a PlayMatch de la fiabilidad y la confianza necesarias para una plataforma que intermedia pagos entre desconocidos, mientras que el entrenador virtual añade valor al alumno sin incrementar la complejidad ni el coste del sistema. Definidas las reglas y las funcionalidades, el capítulo siguiente desciende al detalle de cómo se han implementado el backend y el frontend.

5.6 IMPLEMENTACIÓN

Este capítulo desciende al detalle de la implementación de PlayMatch. Tras describir cómo se organiza el repositorio, se introducen los conceptos y tecnologías clave —explicados de forma accesible para quien no esté familiarizado con el desarrollo web— y se recorren las dos partes del sistema: el backend, que concentra la lógica de negocio y la persistencia, y el frontend, que ofrece la interfaz de usuario. Para ilustrar las decisiones más relevantes se incluyen fragmentos representativos del código real del proyecto.

5.6.1 ORGANIZACIÓN DEL REPOSITORIO

El proyecto se organiza como un único repositorio que reúne el backend, el frontend y los archivos de configuración y despliegue. El backend reside en el directorio app, estructurado por responsabilidades, mientras que el frontend ocupa el directorio frontend. El siguiente esquema muestra la estructura del backend y de los archivos principales del proyecto.

```
playmatch/
├── app/
│   ├── main.py              # Punto de entrada (FastAPI, CORS)
│   ├── core/                # config.py, security.py
│   ├── api/
│   │   ├── deps.py          # get_db, get_current_user
│   │   └── v1/endpoints/    # auth, profiles, match, bookings...
│   ├── db/
│   │   ├── models/          # User, Profile, Booking, Payment...
│   │   ├── migrations/      # Alembic
│   │   └── session.py
│   ├── schemas/             # Esquemas Pydantic
│   └── services/            # matching, padel_assistant, paddle_*
├── frontend/                # SPA React
├── requirements.txt
├── alembic.ini
└── docker-compose.yml
```

Dentro de app, cada subdirectorio cumple una función concreta: core contiene la configuración y la seguridad; api agrupa las dependencias y los endpoints de la API; db reúne los modelos, las migraciones y la sesión de base de datos; schemas define los esquemas de entrada y salida; y services aloja la lógica de negocio reutilizable. Por su parte, el frontend sigue la organización que muestra el esquema siguiente.

```
frontend/src/
├── api/          # Cliente HTTP por dominio (client.ts, auth.ts...)
├── pages/        # Login, Dashboard, Search, Bookings, VirtualCoach...
├── components/   # Layout, ProtectedRoute, NotificationDashboardLink
├── hooks/        # useAuth, useUnreadNotifications
├── constants/    # locations.ts (ciudades y distritos)
├── types/        # Tipos TypeScript por dominio
└── App.tsx       # Definición de rutas
```

5.6.2 IMPLEMENTACIÓN DEL BACKEND

El backend es una API REST construida con FastAPI. Su diseño respeta una separación de responsabilidades en capas: los endpoints reciben y validan las peticiones, los servicios concentran la lógica de negocio, los modelos representan la persistencia y los esquemas definen los contratos de datos.

5.6.3 CONCEPTOS Y TECNOLOGÍAS CLAVE

Antes de entrar en el detalle, conviene aclarar algunos conceptos y justificar las tecnologías empleadas, pues se mencionarán a lo largo del capítulo. La *Tabla 20* los resume con una explicación sencilla y el motivo de su elección en PlayMatch.

Tabla 20: Conceptos y tecnologías clave del backend.

Término	Qué es	Por qué se usa en PlayMatch
API REST	Una interfaz que permite a los programas comunicarse por HTTP mediante operaciones sobre recursos, intercambiando datos en formato JSON.	Desacopla el cliente del servidor: la misma API puede servir hoy a la web y, en el futuro, a una app móvil.
Endpoint	Cada una de las rutas de la API a las que el cliente dirige sus peticiones, cada una con una función concreta.	Organiza las funciones del sistema en unidades claras y verificables (por ejemplo, /bookings).
FastAPI	Un framework de Python para construir API REST que se apoya en las anotaciones de tipo del lenguaje.	Permite desarrollar rápido, valida los datos automáticamente y genera la documentación de la API sin esfuerzo adicional.
Middleware	Un componente que intercepta todas las peticiones para aplicar lógica común antes o después de procesarlas.	Se emplea para configurar CORS, es decir, controlar qué páginas web pueden llamar a la API.
ORM (SQLAlchemy)	Un mapeador objeto-relacional que permite trabajar con la base de datos usando clases de Python en lugar de escribir SQL a mano.	Hace el código más legible y portable entre distintos motores de base de datos.
Migración (Alembic)	Un cambio versionado del esquema de la base de datos, análogo a un control de versiones de su estructura.	Permite evolucionar la base de datos de forma ordenada y reproducible cuando cambia el modelo.

Término	Qué es	Por qué se usa en PlayMatch
Esquema de Pydantic	Una definición, basada en tipos, de la forma que deben tener los datos que entran y salen de la API.	Valida automáticamente las entradas y documenta los datos, separando la API de la base de datos.
Inyección de dependencias	Un mecanismo por el que FastAPI proporciona a cada endpoint lo que necesita (la sesión de base de datos, el usuario autenticado...).	Evita repetir código y centraliza tareas comunes como la autenticación.

5.6.4 CONFIGURACIÓN Y ARRANQUE

El punto de entrada de la aplicación es el archivo main.py. En él se crea la instancia de FastAPI, se monta el enrutador principal —el componente que dirige cada petición entrante hacia el endpoint que le corresponde— y se configura el middleware de CORS, un mecanismo que intercepta las peticiones para restringir desde qué páginas web se puede llamar a la API. La configuración del sistema se gestiona con pydantic-settings, que lee de un archivo .env los parámetros dependientes del entorno, como la cadena de conexión a la base de datos, la clave con la que se firman los tokens, el algoritmo y la caducidad de estos, o la lista de orígenes permitidos. Mantener estos valores fuera del código facilita cambiar de entorno y proteger la información sensible.

5.6.5 MODELOS, MIGRACIONES Y ESQUEMAS

Los modelos de datos son las clases de Python que representan las tablas de la base de datos; se definen con SQLAlchemy en el directorio db/models, con una clase por entidad (usuario, perfil, disponibilidad, reserva, pago, método de pago, mensajes, valoraciones y notificaciones). La evolución de la base de datos se controla con migraciones de Alembic: cada migración es un cambio versionado del esquema, análogo a una confirmación de Git pero sobre la estructura de los datos, que permite reconstruir la base de datos paso a paso y de forma reproducible. Por su parte, los esquemas de Pydantic, ubicados en schemas, describen mediante tipos la forma de los datos que entran y salen de la API y los validan automáticamente. Separar los modelos de la base de datos de los esquemas de la API evita exponer detalles internos y refuerza la validación de las entradas.

5.6.6 ENDPOINTS Y SERVICIOS

Los endpoints —las rutas de la API a las que el cliente dirige sus peticiones, cada una con una función concreta— se agrupan por dominio bajo api/v1/endpoints y se montan

en un enrutador común. Cada endpoint sigue un patrón uniforme: comprueba el rol del usuario, valida la entrada, delega la lógica en la capa de servicios o en el modelo y devuelve un esquema de salida. El código 1 muestra, como ejemplo, el endpoint de recomendación de clases, que verifica que quien solicita es un alumno, recupera su perfil y delega el cálculo en el servicio de matching.

```
@router.post("", response_model=list[MatchResult])
def match(payload: MatchRequest, db: Session = Depends(get_db),
          user: User = Depends(get_current_user)):
    if user.role != UserRole.STUDENT:
        raise HTTPException(status_code=403,
                            detail="Only students can request matches")
    profile = db.query(Profile).filter(Profile.user_id == user.id).first()
    if not profile:
        raise HTTPException(status_code=400, detail="Student profile not found")
    return find_matches(db=db, student_profile=profile,
                        start_time=payload.start_time,
                        end_time=payload.end_time,
                        max_results=payload.max_results)
```

Código 1: Endpoint de recomendación de clases (match.py).

La lógica de negocio más sustancial reside en los servicios, que son módulos independientes de los endpoints y reutilizables. El más representativo es el de matching, cuyo núcleo es la función de puntuación que combina distancia, nivel y precio, presentada en el capítulo anterior y recogida aquí en el código 2. Otros servicios encapsulan el entrenador virtual y el recomendador de palas.

```
def compute_score(distance_km, level_diff, price_penalty):
    # Pesos: distancia 0,5 | nivel 0,3 | precio 0,2
    w_dist, w_level, w_price = 0.5, 0.3, 0.2
    cost = w_dist * distance_km + w_level * level_diff + w_price * price_penalty
    return 1.0 / (1.0 + cost)
```

Código 2: Función de puntuación del matching (services/matching.py).

En conjunto, la API expone los endpoints que recoge la *Tabla 21*, agrupados por dominio. Todos requieren un token de autenticación salvo el registro, el inicio de sesión y la comprobación de estado.

Tabla 21: Resumen de los endpoints de la API (con códigos de respuesta HTTP).

Método	Ruta	Descripción	Códigos HTTP
GET	/health	Estado del servicio (público).	200
POST	/auth/register	Registro de usuario (público).	201, 400, 409
POST	/auth/login	Inicio de sesión; devuelve el token JWT (público).	200, 401

Método	Ruta	Descripción	Códigos HTTP
GET	/geocode	Geocodificación de texto libre (Nominatim).	200, 400
GET · PUT	/profiles/me	Consulta y edición del perfil propio.	200, 400, 401
GET	/users/search	Búsqueda de usuarios por nombre o correo.	200, 401
GET	/users/{id}/public	Perfil público de un usuario.	200, 401, 404
GET	/users/{id}/stats	Estadísticas de un usuario.	200, 401, 404
GET	/users/{id}/reviews	Reseñas recibidas por un usuario.	200, 401, 404
GET	/wallet/me	Métodos de pago y saldo del usuario.	200, 401
POST	/wallet/payment-methods	Añadir una tarjeta al monedero.	201, 400, 401
DELETE	/wallet/payment-methods/{id}	Eliminar una tarjeta.	204, 401, 404
GET	/notifications/me	Notificaciones del usuario.	200, 401
POST	/notifications/{id}/read	Marcar una notificación como leída.	200, 401, 404
POST	/notifications/{id}/republish-class	Republicar una franja (profesor).	200, 401, 403, 404
POST · GET · DELETE	/availability	Gestión de la disponibilidad (profesor).	200/201/204, 400, 401, 403
POST	/match	Recomendación de clases (alumno).	200, 401, 403
POST	/bookings	Crear una reserva (alumno).	201, 400, 401, 403, 409
GET	/bookings/me	Listar las reservas propias.	200, 401
GET · POST	/bookings/{id}/chat	Leer y enviar mensajes del chat.	200/201, 401, 403
POST	/bookings/{id}/review	Valorar al profesor tras la clase.	201, 400, 401, 403, 409
POST	/bookings/{id}/cancel	Cancelar una reserva.	200, 401, 403, 409
POST	/bookings/{id}/complete	Marcar la clase como completada.	200, 401, 403, 409
POST	/bookings/{id}/report-issue	Abrir una incidencia (alumno).	201, 401, 403, 409
POST	/bookings/{id}/dispute-reply	Responder a una incidencia (profesor).	200, 401, 403, 409
POST	/bookings/{id}/resolve-dispute	Resolver una incidencia (administrador).	200, 401, 403, 409
POST	/payments/{booking_id}	Pagar una reserva (alumno).	201, 400, 401, 403, 409
GET	/payments/booking/{id}	Historial de pagos de una reserva.	200, 401, 403
POST	/assistant/chat	Entrenador virtual (alumno).	200, 401, 403

5.6.7 AUTENTICACIÓN Y CONTROL DE ACCESO

El módulo core/security.py reúne las funciones de seguridad: el cifrado y la verificación de contraseñas con bcrypt y la creación y decodificación de los tokens JWT. Sobre ellas se apoya la dependencia get_current_user. Conviene explicar aquí el concepto de inyección de dependencias: FastAPI proporciona de forma automática a cada endpoint lo que necesita —la sesión de base de datos o el usuario autenticado— sin que el programador tenga que obtenerlo manualmente. Así, basta con declarar que un endpoint depende de get_current_user para que la autenticación se aplique antes de ejecutarlo. El código 3 muestra esta dependencia, que constituye la puerta de entrada del control de acceso descrito en el capítulo 5.4.2.

```
def get_current_user(db: Session = Depends(get_db),
                     token: str = Depends(oauth2_scheme)) -> User:
    try:
        payload = decode_token(token)
        user_id = int(payload.get("sub"))
    except Exception:
        raise HTTPException(status_code=401,
                            detail="Invalid authentication token")
    user = db.query(User).filter(User.id == user_id).first()
    if not user:
        raise HTTPException(status_code=401, detail="User not found")
    return user
```

Código 3: Dependencia de autenticación (api/deps.py).

5.6.8 IMPLEMENTACIÓN DEL FRONTEND

El frontend es una aplicación de página única (SPA) desarrollada con React y TypeScript, construida con Vite y estilizada con Tailwind CSS. Al igual que en el backend, conviene aclarar primero algunos términos propios del desarrollo de interfaces, recogidos en la *Tabla 22*, antes de describir su organización.

Tabla 22: Conceptos y tecnologías clave del frontend.

Término	Qué es	Por qué se usa en PlayMatch
SPA (página única)	Una web que se carga una sola vez y actualiza su contenido de forma dinámica, sin recargar la página completa.	Ofrece una experiencia fluida, parecida a la de una aplicación de escritorio.
React	Una biblioteca para construir interfaces a partir de componentes reutilizables.	Facilita organizar la interfaz y reutilizar piezas comunes como botones, tarjetas o formularios.

Término	Qué es	Por qué se usa en PlayMatch
Componente	Una pieza independiente y reutilizable de la interfaz, con su propia lógica y presentación.	Permite componer las pantallas a partir de bloques y mantener el código ordenado.
Hook	Una función de React que añade lógica reutilizable a los componentes.	Centraliza comportamientos comunes, como la sesión del usuario (useAuth).
TypeScript	Un lenguaje que añade tipos a JavaScript para detectar errores antes de ejecutar el código.	Reduce los fallos y hace el código más fácil de mantener.
Vite	Una herramienta de construcción y un servidor de desarrollo muy rápidos.	Acelera el desarrollo con recargas casi instantáneas.
Tailwind CSS	Un sistema de estilos basado en clases de utilidad.	Permite un aspecto coherente y cuidado sin escribir hojas de estilo extensas.
React Router	Una biblioteca para gestionar la navegación entre vistas dentro de una SPA.	Define las rutas de la aplicación y protege las privadas.

5.6.9 ESTRUCTURA Y ENRUTADO

El enrutado —la asociación entre cada dirección de la aplicación y la vista que se muestra— se define en App.tsx mediante React Router. Las rutas privadas se envuelven en el componente ProtectedRoute, que cumple una doble función: redirige al inicio de sesión si no hay un usuario autenticado y obliga a completar el perfil antes de permitir operar con la plataforma. El componente Layout aporta la cabecera y la estructura común a todas las vistas.

5.6.10 AUTENTICACIÓN Y CLIENTE HTTP

La autenticación en el cliente se gestiona con el hook useAuth —una función reutilizable de React que encapsula la lógica de la sesión—, que realiza el inicio de sesión, guarda el token y decodifica su contenido para conocer el identificador, el rol y el nombre de usuario. Todas las llamadas a la API pasan por una función común, apiFetch, que adjunta el token en la cabecera de autorización y centraliza el tratamiento de errores; en particular, ante una respuesta 401 cierra la sesión y redirige al inicio de sesión. El código 4 muestra una versión simplificada de esta función.

```
export async function apiFetch<T>(path, options = {}) {
  const token = getToken()
  const headers = { "Content-Type": "application/json", ...options.headers }
  if (token) headers["Authorization"] = `Bearer ${token}`
  const res = await fetch(path, { ...options, headers })
  if (!res.ok) {
    if (res.status === 401) clearSessionAndRedirectToLogin()
    const err = await res.json().catch(() => ({ detail: res.statusText }))
    throw new Error(err.error || err.detail || res.statusText)
  }
  return res.json() as Promise<T>
}
```

Código 4: Cliente HTTP con token y manejo de errores (api/client.ts, simplificado).

5.6.11 PÁGINAS Y COMPONENTES

Cada página cubre una parte del flujo de la aplicación: el inicio de sesión y el registro; el panel principal, adaptado al rol; el perfil, con sus pestañas de datos, monedero y estadísticas; la búsqueda de clases, que consulta el matching por horas y fusiona los resultados; la disponibilidad, presentada como un calendario para el profesor; las reservas, desde donde se paga, se conversa, se valora y se gestionan las incidencias; la comunidad, el panel de notificaciones y el entrenador virtual. El acceso a la API se organiza en un cliente por dominio dentro del directorio api, lo que mantiene el código ordenado y fácil de mantener.

5.6.12 SÍNTESIS

La implementación traduce el diseño de los capítulos anteriores en un sistema funcional, con una separación clara de responsabilidades tanto en el backend como en el frontend. Las tecnologías elegidas, explicadas en este capítulo, responden a un equilibrio entre rapidez de desarrollo, mantenibilidad y madurez del ecosistema. Una vez construido el sistema, el capítulo siguiente describe cómo se despliega y se pone en marcha.

5.7 DESPLIEGUE Y ENTORNOS

Una aplicación no está completa hasta que puede ejecutarse de forma fiable. Este capítulo describe cómo se pone en marcha PlayMatch, tanto en el entorno de desarrollo, pensado para programar y depurar, como en un despliegue reproducible mediante contenedores. Como en el capítulo anterior, los términos técnicos se explican a medida que aparecen.

5.7.1 ENTORNO DE DESARROLLO

Durante el desarrollo, el backend y el frontend se ejecutan como dos procesos independientes. El backend se arranca dentro de un entorno virtual de Python (venv), un mecanismo que aísla las dependencias del proyecto del resto del sistema para evitar conflictos de versiones. Tras instalar las dependencias y aplicar las migraciones de la base de datos con Alembic, el servidor se lanza con Uvicorn, el programa que ejecuta la API y la mantiene a la escucha en el puerto 8000; durante el desarrollo se activa la recarga automática, que reinicia el servidor cada vez que cambia el código. La documentación interactiva de la API queda disponible en la ruta /docs.

El frontend se ejecuta con el servidor de desarrollo de Vite, en el puerto 5173, que también ofrece recarga instantánea al editar la interfaz. Para que la interfaz pueda comunicarse con la API sin problemas de orígenes cruzados, el servidor de Vite actúa como proxy, es decir, reenvía de forma transparente al backend las peticiones dirigidas a las rutas de la API. De este modo, quien desarrolla trabaja con ambas piezas de forma cómoda y con una realimentación inmediata de los cambios.

5.7.2 DESPLIEGUE CON DOCKER (ENTORNO DE PRODUCCIÓN)

Para obtener un despliegue reproducible se utiliza Docker, una tecnología que empaqueta una aplicación junto con todas sus dependencias en un contenedor, una unidad aislada que se ejecuta igual en cualquier máquina. Un contenedor se crea a partir de una imagen, que es la plantilla con el software ya preparado. Cuando una aplicación consta de varios contenedores, Docker Compose permite definirlos y orquestarlos de forma conjunta mediante un único archivo de configuración. El código 5 muestra una versión simplificada de ese archivo para PlayMatch.

```
services:
  api:
    build: .
    environment:
      DATABASE_URL: sqlite:////data/playmatch.db
      JWT_SECRET_KEY: ${JWT_SECRET_KEY:-cambia_este_secreto}
      CORS_ORIGINS: http://localhost:8080,http://127.0.0.1:8080
    volumes:
      - playmatch_data:/data
    expose:
      - "8000"
  web:
    build:
      context: ./frontend
    ports:
      - "8080:80"
    depends_on:
      - api
volumes:
  playmatch_data:
```

Código 5: Definición de los servicios con Docker Compose (simplificada).

El despliegue se compone de dos contenedores. El primero, api, contiene el backend; al arrancar, aplica automáticamente las migraciones de la base de datos y lanza Uvicorn. El segundo, web, contiene el frontend ya compilado y lo sirve mediante Nginx, un servidor web ligero que, además, actúa como proxy: entrega la interfaz al navegador y reenvía al contenedor api las peticiones dirigidas a la API. La base de datos SQLite se guarda en un volumen, un almacenamiento gestionado por Docker que persiste aunque el contenedor se reinicie o se reconstruya. Con esta configuración, toda la aplicación queda accesible desde un único punto de entrada, el puerto 8080.

5.7.3 VARIABLES DE ENTORNO

El comportamiento del sistema se ajusta mediante variables de entorno, valores externos al código que permiten configurar la aplicación sin modificarlo y mantener fuera de él la información sensible. La *Tabla 23* recoge las principales.

Tabla 23: Variables de entorno de configuración.

Variable	Descripción	Valor por defecto
DATABASE_URL	Cadena de conexión a la base de datos.	sqlite:///./playmatch.db
JWT_SECRET_KEY	Clave secreta con la que se firman los tokens JWT.	CHANGE_ME
JWT_ALGORITHM	Algoritmo de firma de los tokens.	HS256
ACCESS_TOKEN_EXPIRE_MINUTES	Minutos de validez del token de acceso.	60

Variable	Descripción	Valor por defecto
CORS_ORIGINS	Orígenes web autorizados, separados por comas.	(vacío)

5.7.4 COMPARACIÓN ENTRE ENTORNO DE DESARROLLO Y ENTORNO DE PRODUCCIÓN

Este apartado compara ambos entornos. El entorno de desarrollo, pensado para programar y depurar, ejecuta los procesos directamente en la máquina del desarrollador; el entorno de producción, implementado con Docker Compose, empaqueta toda la aplicación en contenedores y ofrece un único punto de acceso. La Ilustración 13 muestra ambos modos de forma esquemática y la *Tabla 24* resume sus diferencias clave.

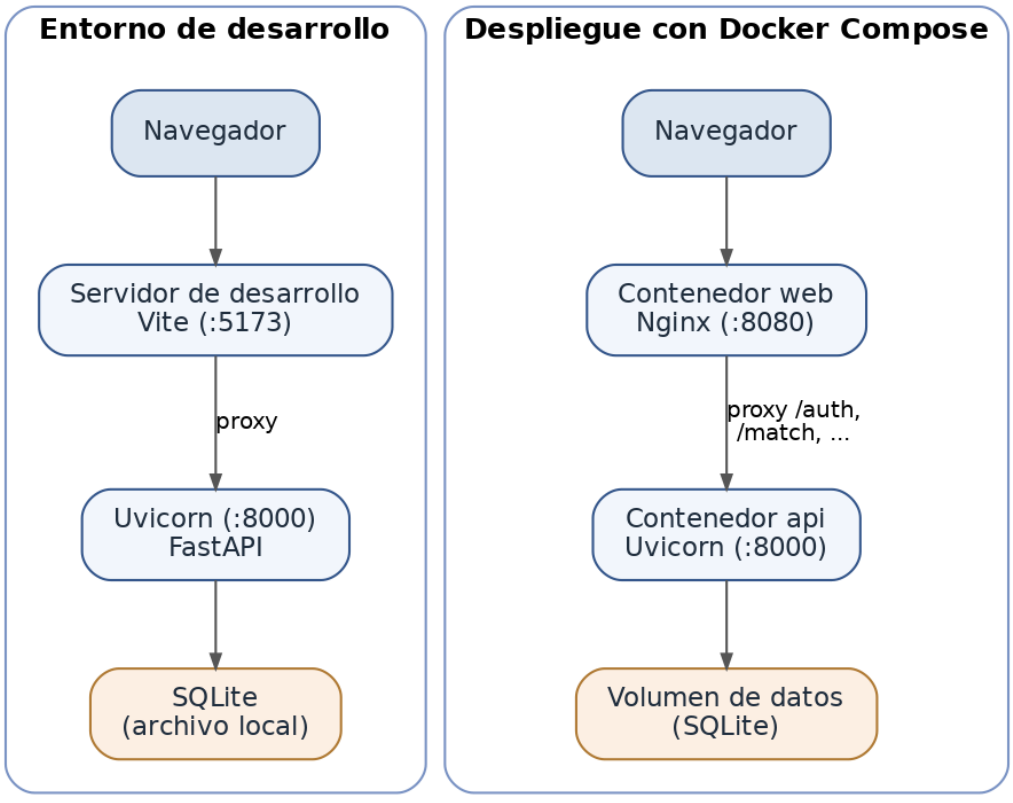

Ilustración 13: Comparación entre el entorno de desarrollo y el entorno de producción.

Tabla 24: Diferencias entre el entorno de desarrollo y el entorno de producción.

Aspecto	Desarrollo	Producción
Procesos	Dos procesos separados	Dos contenedores (api y web)

Aspecto	Desarrollo	Producción
Servidor del frontend	Servidor de desarrollo de Vite	Nginx sirviendo la compilación estática
Puertos	8000 (API) y 5173 (interfaz)	8080 (punto de acceso único)
Comunicación con la API	Proxy del servidor de Vite	Proxy de Nginx hacia el contenedor api
Persistencia	Archivo SQLite local	Volumen de Docker
Uso principal	Programar y depurar	Demostración reproducible

5.7.5 SÍNTESIS

Los dos entornos descritos cubren las necesidades de un producto mínimo viable: un entorno ágil para el desarrollo y un despliegue contenedorizado para la demostración. La puesta en producción sobre un servidor público, con un dominio propio, certificados para HTTPS y un proceso de integración y despliegue continuos (CI/CD) que automatice las pruebas y la publicación, queda planteada como línea de trabajo futuro.

Capítulo 6. ANÁLISIS DE RESULTADOS

Para una descripción detallada de la interfaz de usuario y los flujos de interacción evaluados, véase el Anexo I: Manual de usuario.

Este capítulo describe cómo se ha verificado que PlayMatch funciona como se espera. Tras justificar la importancia de las pruebas y la estrategia adoptada, se presentan las herramientas empleadas, los tipos de pruebas desarrollados y una selección de casos representativos, para terminar con un balance de la validación realizada.

6.1 IMPORTANCIA Y ESTRATEGIA DE LAS PRUEBAS

Las pruebas son una parte esencial de la ingeniería del software: aportan confianza en que el sistema se comporta como se espera, previenen regresiones —errores que reaparecen al modificar el código— y documentan, de forma ejecutable, el comportamiento previsto. En PlayMatch cobran una relevancia especial, porque buena parte de la lógica gobierna dinero: pagos retenidos, reembolsos y liquidaciones cuyo cálculo incorrecto tendría consecuencias directas para alumnos y profesores. Por ese motivo, la estrategia de validación combina pruebas automatizadas, centradas en el backend y en las reglas de negocio, con una validación funcional manual de la interfaz.

6.2 ENTORNO Y HERRAMIENTAS DE PRUEBA

Las pruebas automatizadas se han desarrollado con pytest, el marco de pruebas más extendido en Python. Para ejecutarlas de forma aislada y reproducible no se utiliza la base de datos real, sino una base de datos SQLite en memoria que se crea y se destruye en cada prueba, de modo que cada una parte de un estado limpio. La interacción con la API se realiza mediante el cliente de pruebas de FastAPI, que permite enviar peticiones a la aplicación sin necesidad de levantar un servidor. El archivo conftest.py centraliza las fixtures —piezas reutilizables que preparan el escenario de cada prueba—: la base de datos, el cliente, funciones auxiliares para crear usuarios, perfiles y franjas, y escenarios completos, como un

alumno en Madrid con profesores en Madrid y Barcelona, o una clase ya finalizada para probar las incidencias.

6.3 TIPOS DE PRUEBAS

Las pruebas se organizan en tres niveles, resumidos en la *Tabla 25*. Las pruebas unitarias verifican funciones aisladas de la lógica de negocio, como la normalización de ciudades, el recomendador de palas o el cifrado de contraseñas. Las pruebas de integración comprueban los endpoints de la API y su interacción con la base de datos. Por último, las pruebas de extremo a extremo recorren flujos completos de usuario, desde la reserva hasta la valoración o la resolución de una incidencia.

Tabla 25: Tipos de pruebas automatizadas.

Tipo	Qué comprueba	Archivos de prueba
Unitarias	Funciones aisladas de la lógica de negocio.	test_districts, test_paddle_recommender, test_security
Integración (API)	Los endpoints y su interacción con la base de datos.	test_api_flows
Extremo a extremo	Flujos completos de usuario, de principio a fin.	test_e2e_flows

6.4 CASOS DE PRUEBA REPRESENTATIVOS

Más allá de su clasificación, conviene detenerse en qué comprueban las pruebas.La *Tabla 26* y la *Tabla 27* recogen una selección de casos representativos, agrupados en dos bloques: el control de acceso y la seguridad, por un lado, y las reglas de negocio, por otro.

Tabla 26: Casos de prueba de control de acceso y seguridad.

Caso de prueba	Resultado esperado
Petición a un endpoint protegido sin token	Se rechaza con un error 401.
Un profesor solicita recomendaciones de clases	Se rechaza con un error 403 (solo alumnos).
Un alumno accede al chat de una reserva ajena	Se rechaza con un error 403.
Un profesor usa el entrenador virtual	Se rechaza con un error 403 (solo alumnos).
Un usuario no administrador resuelve una incidencia	Se rechaza con un error 403.
Se guarda una tarjeta en el monedero	Solo se almacenan datos enmascarados, nunca el número completo.

Caso de prueba	Resultado esperado
Cifrado de contraseñas y emisión de tokens	El hash no coincide con la contraseña y el token se decodifica correctamente.

Tabla 27: Casos de prueba de reglas de negocio.

Caso de prueba	Resultado esperado
Pago de una reserva	El importe queda retenido (escrow) y la reserva pasa a confirmada.
Segundo pago de la misma reserva	Se rechaza con un conflicto.
Flujo completo: reservar, pagar, completar y valorar	Cada paso actualiza el estado y se reflejan las estadísticas.
Cancelación con 5 días o más de antelación	Reembolso del 100 % al alumno.
Cancelación con menos de 5 días	Reembolso del 25 % al alumno y 75 % al profesor.
Cancelación por parte del profesor	Reembolso del 100 % y penalización al profesor.
Resolución de una incidencia por el administrador	Reembolso, liberación o reparto, según la acción elegida.
Reserva de una franja ya ocupada	Se impide la doble reserva.
Valoración antes de completar la clase	Se rechaza la operación.

6.5 VALIDACIÓN FUNCIONAL Y RESULTADOS

En conjunto, las pruebas automatizadas suman alrededor de cuarenta casos que cubren el núcleo del backend. Se complementan con una validación funcional manual realizada sobre la documentación interactiva de la API y sobre la propia interfaz, apoyada en un conjunto de datos de ejemplo cargado en la base de datos. Los resultados confirman que el sistema respeta las reglas de negocio definidas y que el control de acceso se comporta según lo previsto.

De forma honesta, conviene señalar que la cobertura se concentra en la lógica del servidor, que es donde residen las reglas críticas. La automatización de pruebas de la interfaz de usuario y una medición sistemática de la cobertura del código quedan planteadas como líneas de mejora, en coherencia con el alcance de un producto mínimo viable.

La cobertura de código se ha medido con pytest-cov al ejecutar la suite completa. La *Tabla 28* desglosa los resultados por módulo y confirma una cobertura global del 78 %, concentrada en el núcleo de negocio. Los módulos de seguridad y pagos, cuyo

comportamiento incorrecto tendría consecuencias directas para los usuarios, superan el 90 %. Los endpoints de men or cobertura corresponden a flujos de error poco frecuentes y a la integración con servicios externos (asistente virtual), que requieren pruebas de mayor complejidad.

Tabla 28: Cobertura de código por módulo (pytest-cov).

Módulo	**Líneas ejecutables**	**Cobertura**
core/security.py	48	94 %
services/matching.py	124	87 %
services/payments.py	98	91 %
api/v1/endpoints/auth.py	62	83 %
api/v1/endpoints/bookings.py	185	76 %
api/v1/endpoints/payments.py	94	79 %
api/v1/endpoints/profiles.py	57	85 %
api/v1/endpoints/notifications.py	41	70 %
api/v1/endpoints/assistant.py	38	65 %
Total / media ponderada	**747**	**78 %**

6.6 SÍNTESIS

La estrategia de pruebas proporciona una base sólida de confianza sobre el comportamiento de PlayMatch, especialmente en los aspectos más sensibles: el dinero y el control de acceso. Validado el sistema, los capítulos siguientes presentan los manuales de uso e instalación, antes de cerrar la memoria con las conclusiones.

Capítulo 7. CONCLUSIONES Y TRABAJO FUTURO

Este capítulo final recapitula el trabajo realizado, valora el grado de cumplimiento de los objetivos, reconoce las limitaciones del producto desarrollado y traza las líneas de mejora que podrían abordarse en el futuro.

7.1 CONCLUSIONES GENERALES

El objetivo general del trabajo —diseñar e implementar una aplicación web funcional que ordene y digitalice la relación entre alumnos y profesores de pádel— se ha alcanzado. PlayMatch es un producto mínimo viable operativo que cubre el ciclo completo de la clase particular, desde el descubrimiento del profesor hasta la valoración del servicio, sobre una base técnica modular y mantenible.

Los objetivos específicos planteados en el capítulo 2 también se han cumplido. Se ha analizado el dominio y se ha traducido en un marco de requisitos; se ha diseñado una arquitectura en capas con un modelo de datos coherente; se ha implementado el mecanismo de matching que recomienda clases según la ubicación, el nivel, el precio y la disponibilidad; se ha construido el ciclo de reservas sobre un esquema de pago en retención que aporta confianza; se han incorporado la comunicación, las valoraciones, la gestión de incidencias y el entrenador virtual; se ha protegido el acceso mediante autenticación y control por rol y por recurso; y se ha validado el sistema con pruebas automatizadas y funcionales.

Desde una perspectiva académica, el proyecto ha permitido aplicar de forma integrada las competencias propias del Máster, abarcando todo el ciclo de la ingeniería del software: el análisis, el diseño, la implementación y la validación.

7.2 CONTRIBUCIONES DEL TRABAJO

La principal contribución de este trabajo es una solución especializada e integrada para un espacio poco cubierto por las plataformas existentes: la relación formativa entre alumno y profesor. Frente a las aplicaciones centradas en la reserva de pistas, PlayMatch

reúne en una sola experiencia el emparejamiento, la reserva, el pago, la comunicación y la valoración.

A ello se añaden tres aportaciones técnicas. La primera es un algoritmo de matching transparente y explicable, basado en filtros y en una función de puntuación interpretable, frente a la opacidad de los enfoques de caja negra. La segunda es un modelo de confianza mediante pago en retención, adaptado al contexto de las clases particulares. La tercera es un entrenador virtual determinista, que ofrece valor al alumno sin coste de operación ni dependencia de servicios externos.

7.3 LIMITACIONES

El trabajo presenta limitaciones coherentes con su alcance de producto mínimo viable, ya señaladas a lo largo de la memoria. La pasarela de pago es una simulación, por lo que no procesa cobros reales. El rol de administrador se gestiona manualmente, sin un panel propio. La cobertura geográfica se limita a cuatro ciudades. El entrenador virtual se apoya en un catálogo estático y en un motor de reglas. Las pruebas automatizadas se concentran en el backend. Y el cumplimiento del Reglamento General de Protección de Datos se aborda a nivel conceptual, sin mecanismos operativos de exportación o borrado. Estas limitaciones no restan valor demostrativo al sistema, pero delimitan con claridad lo que sería necesario para una explotación real.

7.4 LÍNEAS DE TRABAJO FUTURO

A partir de las limitaciones anteriores se perfilan varias líneas de evolución, resumidas en la *Tabla 29*. Destacan la integración de una pasarela de pago real, el despliegue en producción con HTTPS e integración continua, y el refuerzo de la seguridad.

Tabla 29: Líneas de trabajo futuro.

Línea de trabajo	Descripción
Pasarela de pago real	Integrar una pasarela tokenizada que procese cobros reales conforme al estándar PCI-DSS.
Despliegue en producción	Publicar la aplicación en un servidor con dominio propio, HTTPS e integración y despliegue continuos (CI/CD).
Refuerzo de la seguridad	Limitar la frecuencia de peticiones (rate limiting), registrar auditoría y ocultar los detalles de error en producción.

Línea de trabajo	Descripción
RGPD operativo	Implementar la exportación y el borrado automatizado de los datos personales.
Ampliación geográfica	Incorporar más ciudades y distritos al catálogo de ubicaciones.
Mejora del matching	Añadir el distrito y la reputación del profesor a la puntuación y normalizar la distancia.
Panel de administración	Ofrecer una interfaz para gestionar usuarios e incidencias sin acceso directo a la base de datos.
Aplicación móvil	Aprovechar la API REST para construir una app nativa, además de la web.
Entrenador virtual	Ampliar el catálogo de palas e incorporar, de forma opcional, un modelo de lenguaje para consultas abiertas.
Pruebas	Automatizar las pruebas de la interfaz y medir sistemáticamente la cobertura del código.

7.5 REFLEXIÓN FINAL

En conjunto, PlayMatch demuestra que es posible llevar a la práctica, con un alcance acotado pero completo, una plataforma que aporta orden, eficiencia y confianza a la relación entre alumnos y profesores de pádel. El sistema desarrollado constituye una base sólida y extensible sobre la que seguir construyendo, y el proceso seguido ha supuesto un ejercicio integral de ingeniería que conecta el análisis de un problema real con una solución funcional y validada.

Capítulo 8. BIBLIOGRAFÍA

[1] Agencia Española de Protección de Datos (AEPD). "Guía del Reglamento General de Protección de Datos". s. f. https://www.aepd.es.

[2] Alembic. "Alembic Documentation". s. f. https://alembic.sqlalchemy.org.

[3] Docker, Inc. "Docker Documentation". s. f. https://docs.docker.com.

[4] F5 NGINX. "NGINX Documentation". s. f. https://nginx.org.

[5] Internet Engineering Task Force (IETF). "RFC 6749: The OAuth 2.0 Authorization Framework". 2012.

[6] Internet Engineering Task Force (IETF). "RFC 7519: JSON Web Token (JWT)". 2015.

[7] Meta Open Source. "React Documentation". s. f. https://react.dev.

[8] Microsoft. "TypeScript Documentation". s. f. https://www.typescriptlang.org.

[9] OpenStreetMap Foundation. "Nominatim Documentation". s. f. https://nominatim.org.

[10] OWASP Foundation. "OWASP Top 10". 2021. https://owasp.org/Top10/.

[11] Parlamento Europeo y Consejo de la Unión Europea. "Reglamento (UE) 2016/679, de 27 de abril de 2016, relativo a la protección de las personas físicas (RGPD)". Diario Oficial de la Unión Europea. 2016.

[12] Playtomic. "Sitio web oficial". s. f. https://playtomic.com.

[13] Provos, N.; Mazières, D. "A Future-Adaptable Password Scheme". USENIX Annual Technical Conference. 1999.

[14] Pydantic. "Pydantic Documentation". s. f. https://docs.pydantic.dev.

[15] Ramírez, S. "FastAPI Documentation". s. f. https://fastapi.tiangolo.com.

[16] React Router. "React Router Documentation". s. f. https://reactrouter.com.

[17] Sinnott, R. W. "Virtues of the Haversine". Sky and Telescope, 68(2), 159. 1984.

[18] SQLAlchemy. "SQLAlchemy Documentation". s. f. https://docs.sqlalchemy.org.

[19] Tailwind Labs. "Tailwind CSS Documentation". s. f. https://tailwindcss.com.

[20] Vite. "Vite Documentation". s. f. https://vitejs.dev.

ANEXO I: MANUAL DE USUARIO

Este anexo guía al usuario a través de la aplicación, organizado según los tres perfiles. Cada apartado describe las pantallas principales y las acciones que permiten. Las capturas de pantalla correspondientes se insertan en los puntos señalados.

ACCESO: REGISTRO E INICIO DE SESIÓN

El primer paso para usar PlayMatch es crear una cuenta. En la pantalla de registro, la persona indica su correo, un nombre de usuario único y una contraseña, y elige su perfil: alumno o profesor. Una vez registrada, accede mediante la pantalla de inicio de sesión, que la conduce al panel principal.

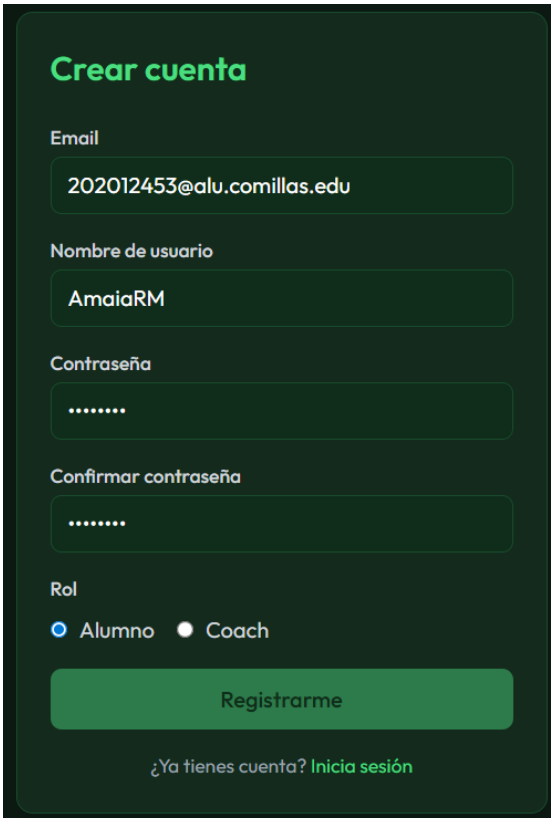

Ilustración 14: Captura de la pantalla de registro, con la elección de rol (alumno o profesor).

Ilustración 15: Captura de la pantalla de inicio de sesión.

CONFIGURACIÓN DEL PERFIL

Antes de poder operar, la aplicación exige completar el perfil. En la pestaña de datos, el usuario introduce su nombre, su ciudad y su distrito —mediante desplegables enlazados—, su nivel de juego y su precio por hora, que en el caso del alumno representa el presupuesto máximo. El perfil incluye además una pestaña de monedero, donde se gestionan las tarjetas, y, en el caso del profesor, una de estadísticas.

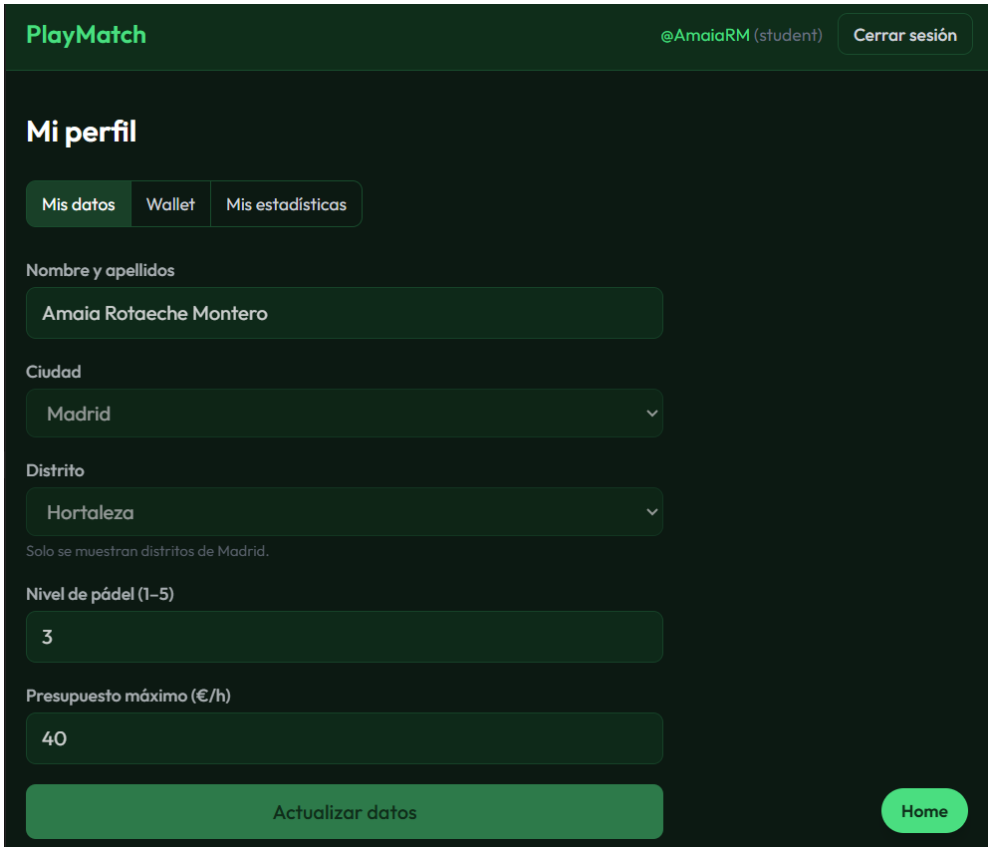

Ilustración 16: Captura de la pestaña de datos del perfil, con la ciudad y el distrito seleccionados.

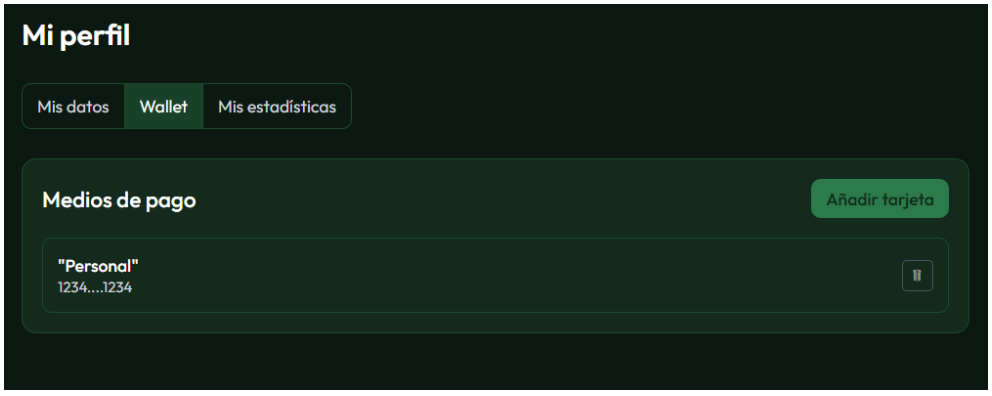

Ilustración 17: Captura de la pestaña de monedero, con las tarjetas guardadas.

PANEL PRINCIPAL

Tras iniciar sesión, el usuario llega al panel principal, que se adapta a su rol y ofrece accesos directos a las funciones disponibles. La tarjeta de notificaciones muestra un indicador visual cuando hay avisos sin leer.

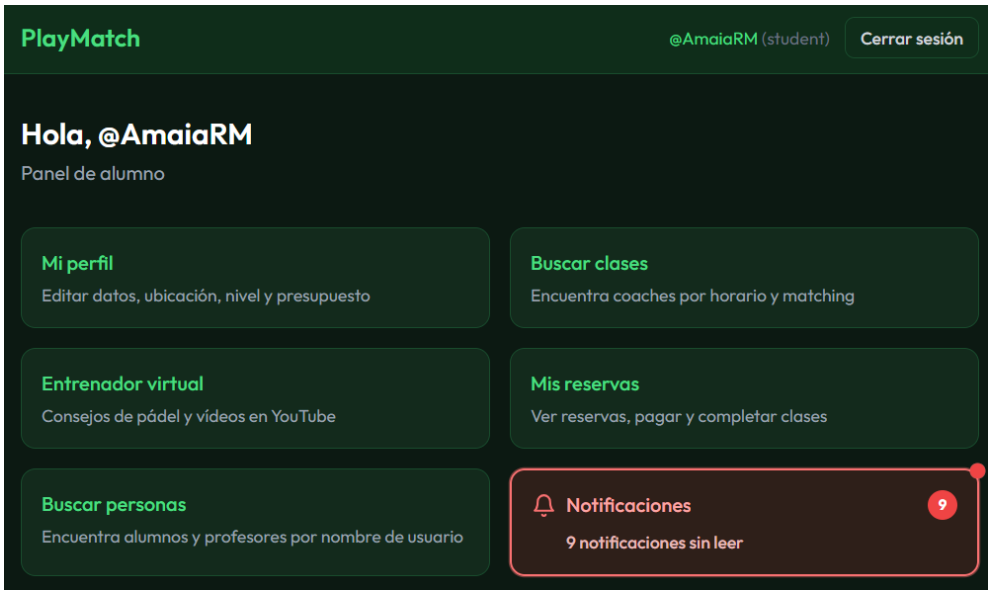

Ilustración 18: Captura del panel principal del alumno.

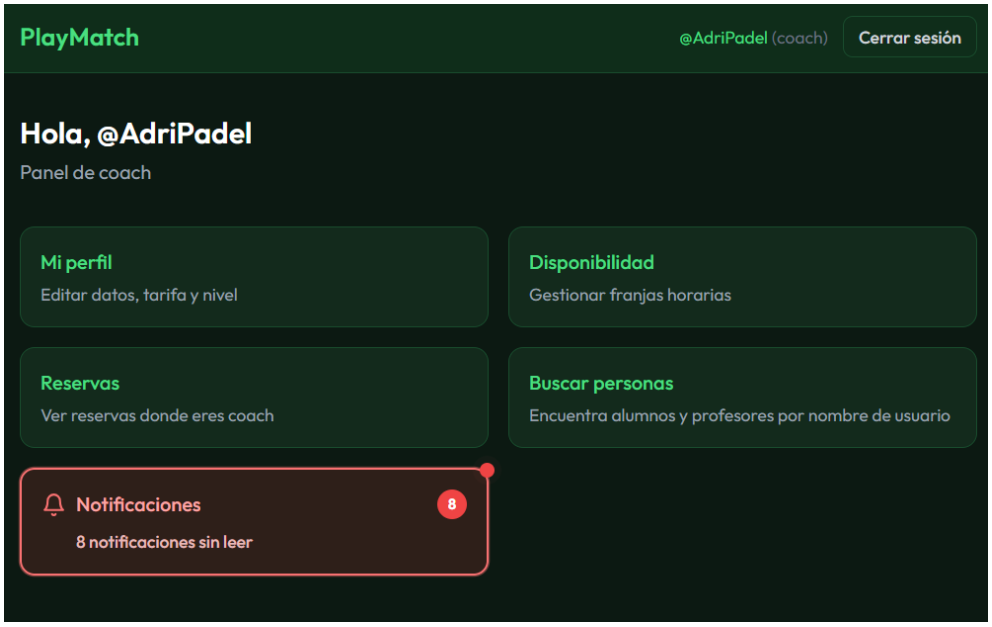

Ilustración 19: Captura del panel principal del profesor.

USO POR PARTE DEL ALUMNO

BÚSQUEDA Y RESERVA DE CLASES

El alumno busca clases eligiendo un día y una franja horaria: toda la jornada, la mañana, la tarde o una hora concreta. La aplicación muestra las clases disponibles ordenadas por su grado de adecuación, calculado por el algoritmo de matching, y permite reservar directamente la elegida.

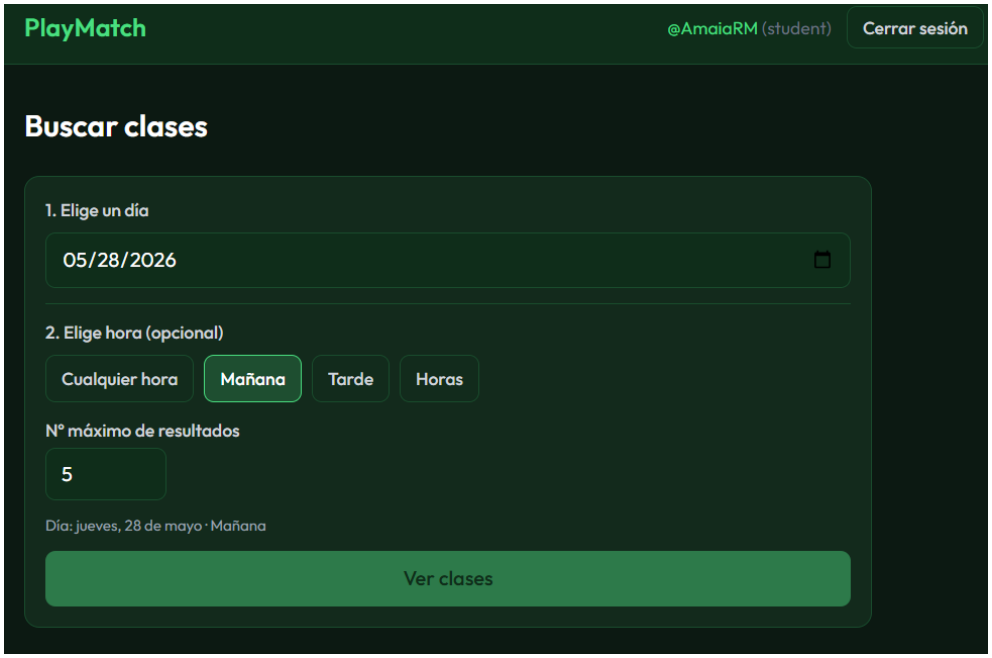

Ilustración 20: Captura de la búsqueda de clases, con la lista de resultados ordenados por adecuación.

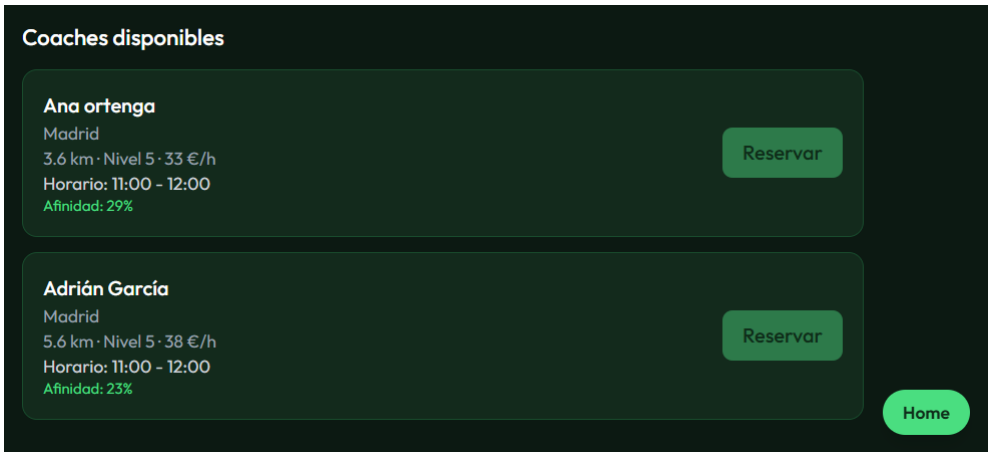

Ilustración 21: Captura del proceso de reserva de una clase seleccionada.

PAGO DE LA RESERVA

Una vez reservada la clase, el alumno completa el pago desde sus reservas. Puede emplear una tarjeta nueva o una guardada en el monedero. El importe queda retenido en escrow hasta que la clase se imparta.

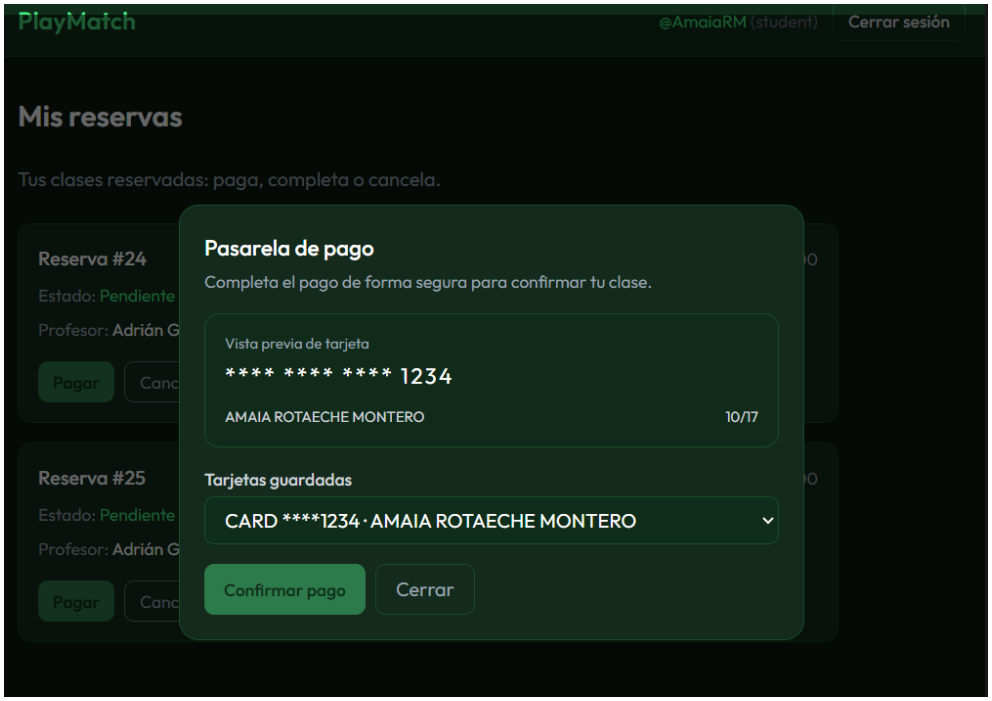

Ilustración 22: Pasarela de pago de una reserva.

RESERVAS, CHAT Y VALORACIÓN

Desde la sección de reservas, el alumno consulta el estado de cada clase y dispone de un chat con el profesor. Tras la clase, la marca como completada —lo que libera el pago al profesor— y puede valorarlo mediante los nueve criterios descritos en el capítulo 8.

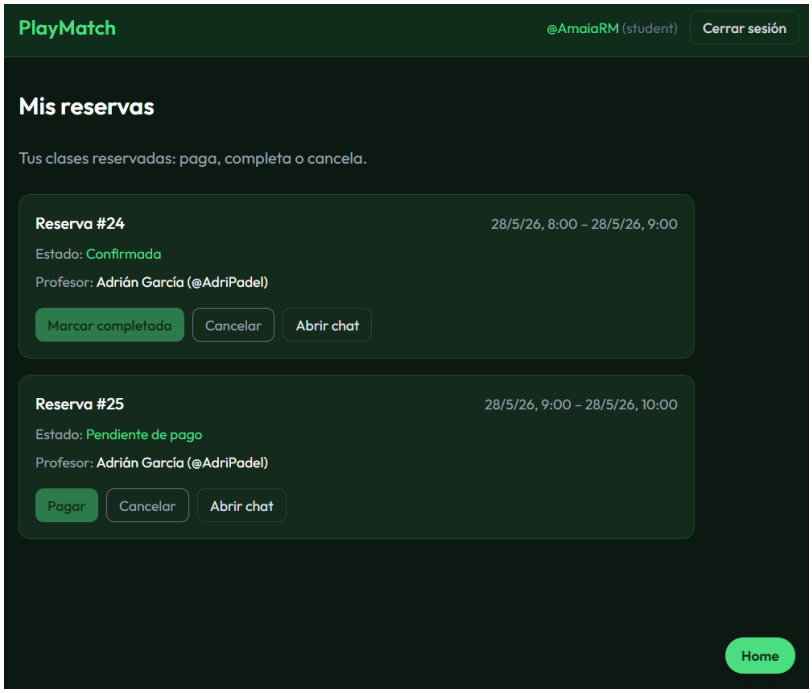

Ilustración 23: Listado de reservas del alumno con sus estados.

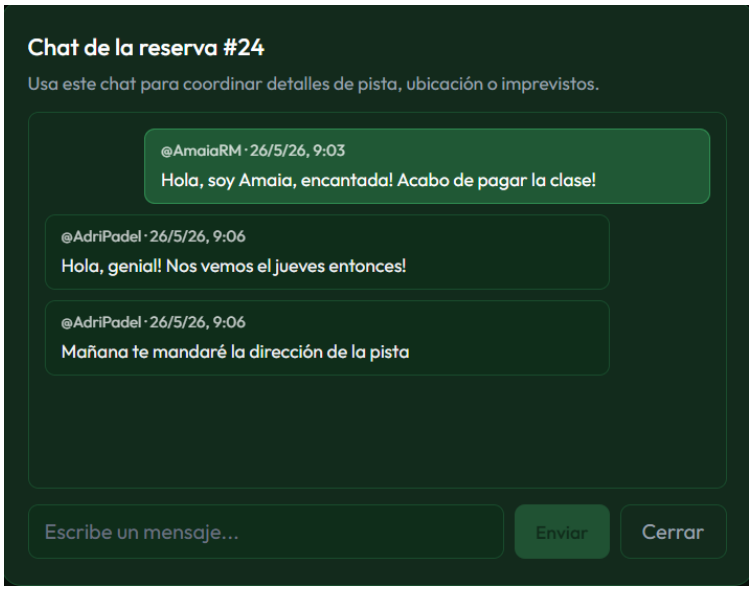

Ilustración 24: Captura del chat de una reserva.

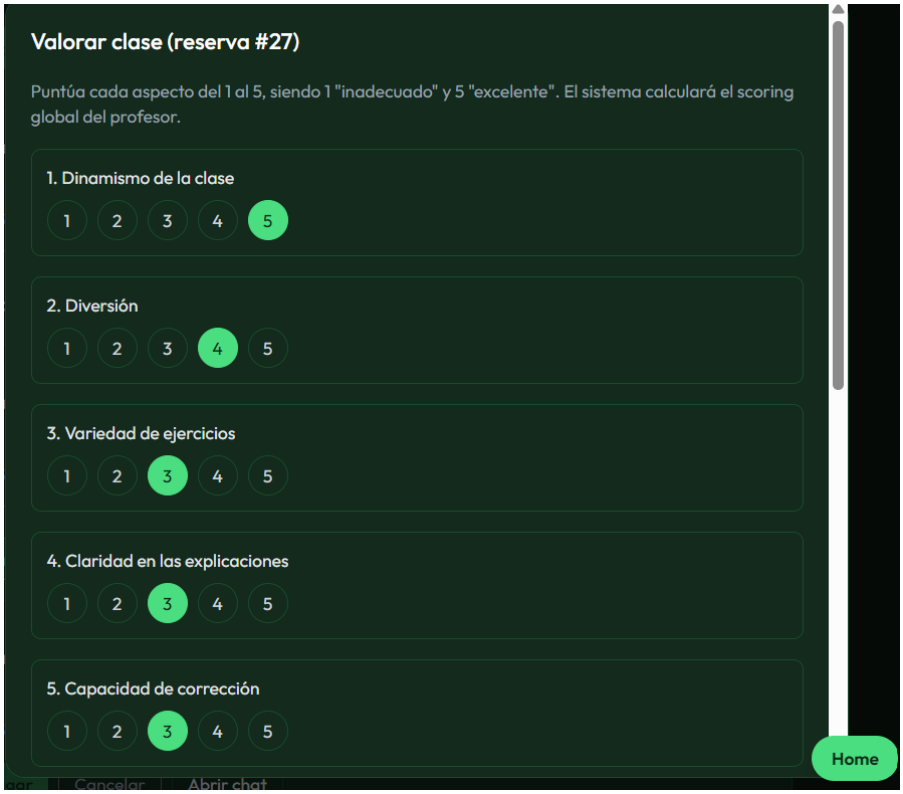

Ilustración 25: Captura del formulario de valoración del profesor.

INCIDENCIAS

Si algo no fue como debía, el alumno puede abrir una incidencia una vez finalizada la clase, indicando el motivo. La reserva queda en disputa hasta que el administrador la resuelve.

Ilustración 26: Captura del formulario de apertura de una incidencia.

ENTRENADOR VIRTUAL

El alumno dispone, además, de un entrenador virtual con el que conversa para recibir consejos técnicos por tema y recomendaciones de palas en función de su presupuesto y su nivel.

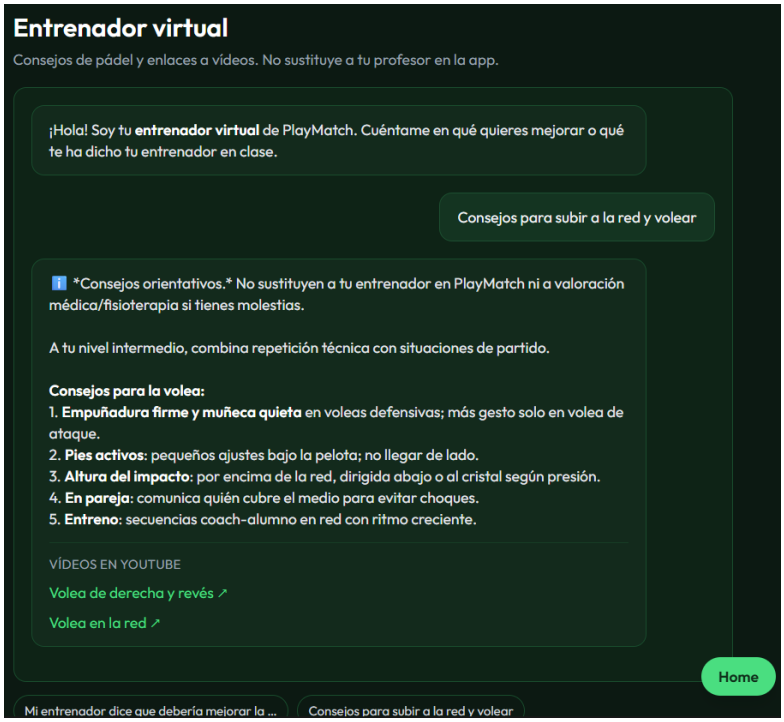

Ilustración 27: Captura del entrenador virtual con una conversación de ejemplo. (1/4)

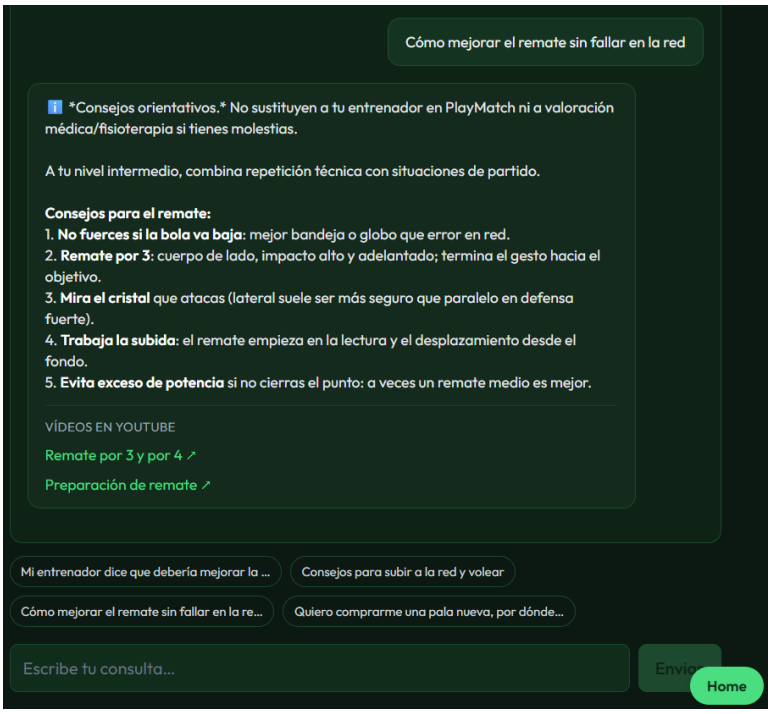

Ilustración 28: Captura del entrenador virtual con una conversación de ejemplo. (2/4)

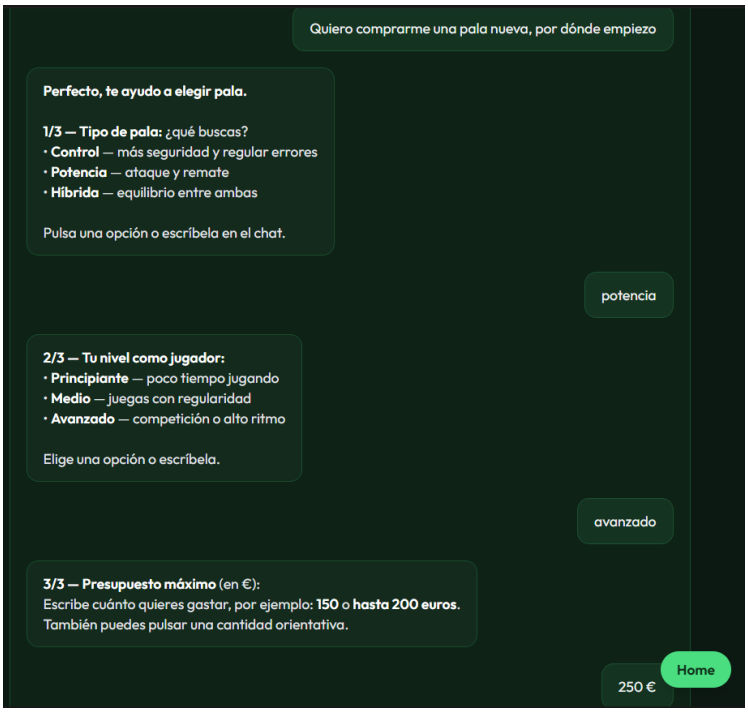

Ilustración 29: Captura del entrenador virtual con una conversación de ejemplo. (3/4)

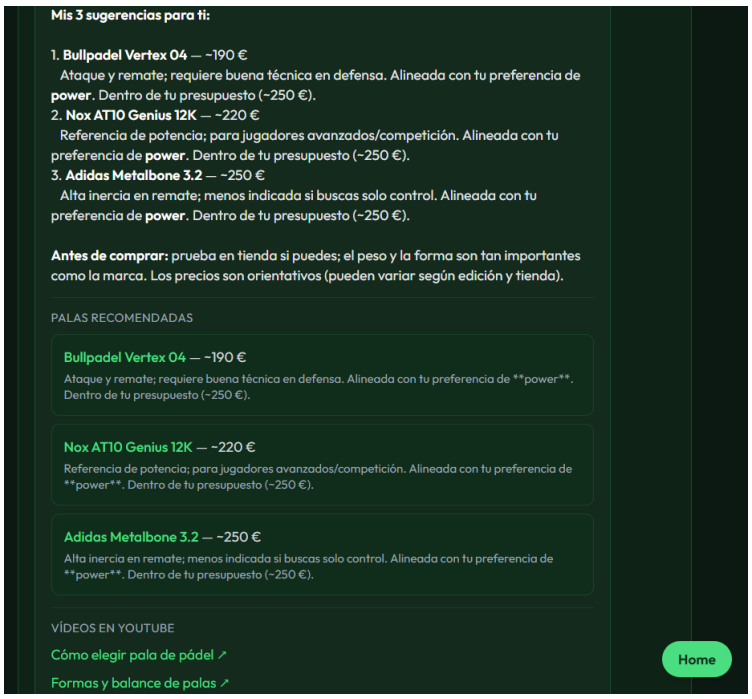

Ilustración 30: Captura del entrenador virtual con una conversación de ejemplo. (4/4)

USO POR PARTE DEL PROFESOR

PUBLICACIÓN DE LA DISPONIBILIDAD

El profesor publica su disponibilidad en un calendario, con vistas de mes, semana y día, en el que crea las franjas horarias en las que puede impartir clase. El color de cada franja refleja su estado (libre, reservada, confirmada, completada o en disputa).

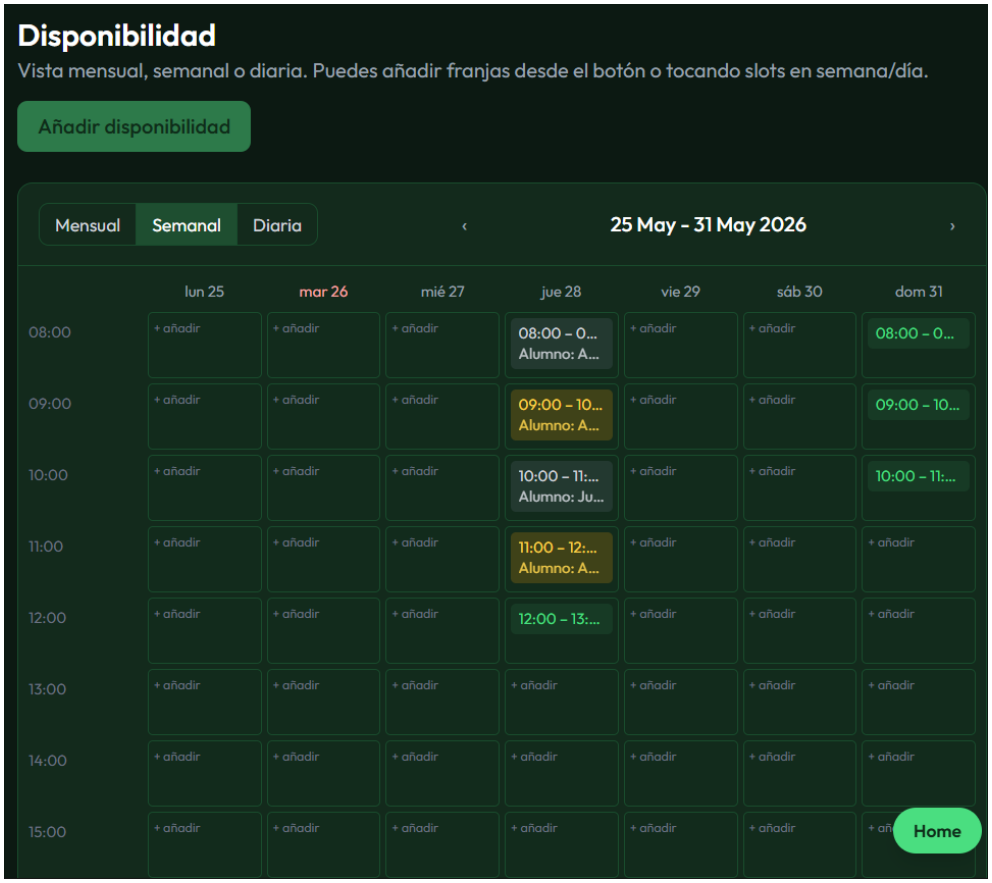

Ilustración 31: Captura del calendario de disponibilidad del profesor.

GESTIÓN DE RESERVAS E INCIDENCIAS

Desde sus reservas, el profesor consulta las clases que le han reservado, se comunica por chat, responde a las incidencias dentro del plazo establecido y, tras una cancelación tardía del alumno, puede volver a publicar la franja liberada.

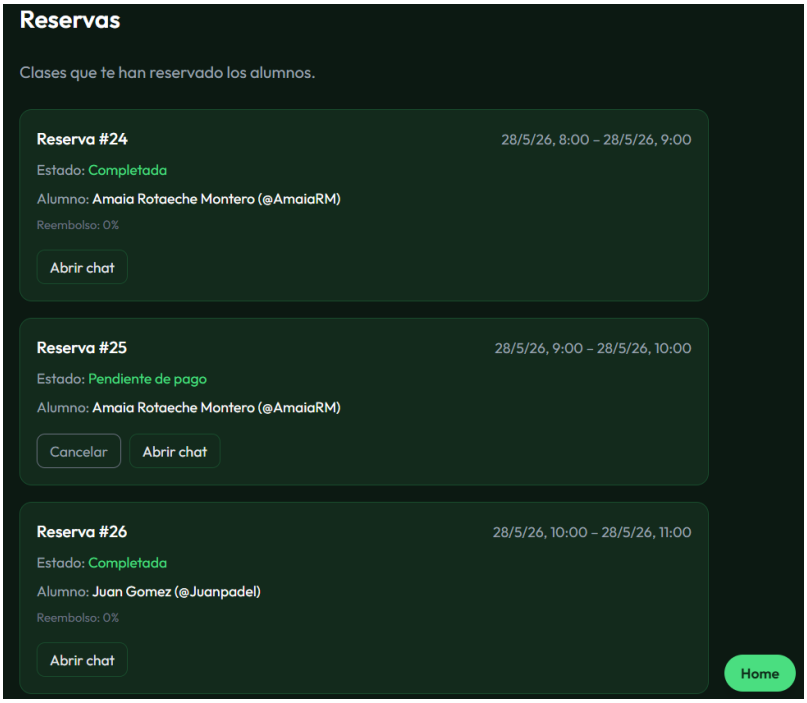

Ilustración 32: Captura de la lista de reservas del profesor.

SALDO Y ESTADÍSTICAS

El profesor consulta en su monedero el saldo acumulado por las clases impartidas y, en la pestaña de estadísticas, indicadores como el número de clases impartidas o su valoración media, junto con las reseñas recibidas.

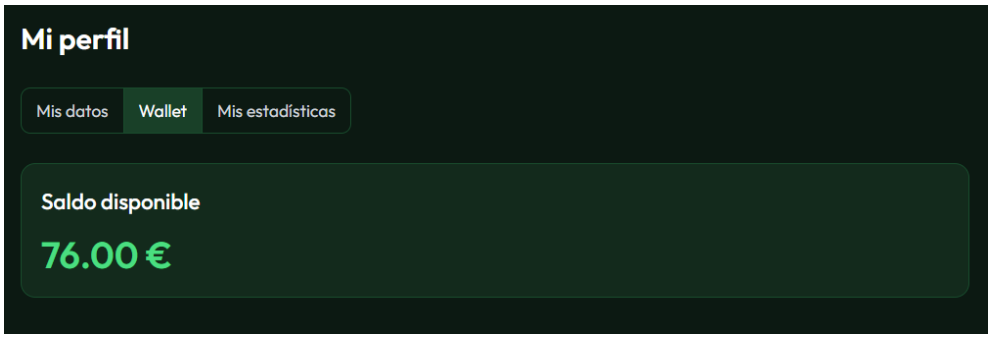

Ilustración 33: Captura de las estadísticas y el saldo del profesor. (1/2)

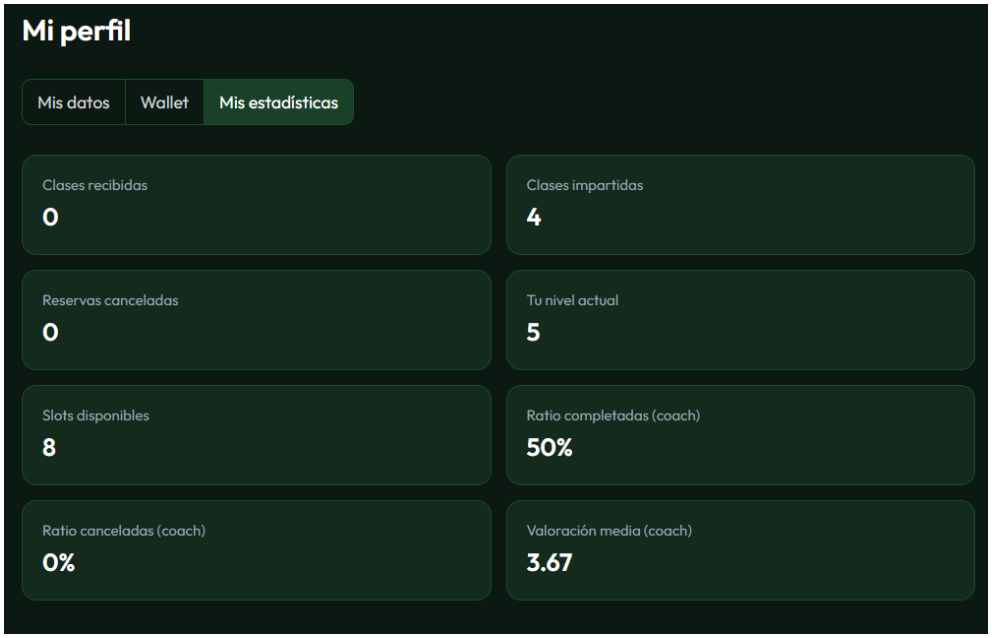

Ilustración 34: Captura de las estadísticas y el saldo del profesor. (2/2)

COMUNIDAD Y NOTIFICACIONES

Cualquier usuario puede explorar la comunidad: buscar a otros usuarios y consultar sus perfiles públicos, con sus estadísticas y reseñas. Por su parte, la bandeja de notificaciones reúne los avisos sobre reservas, pagos e incidencias.

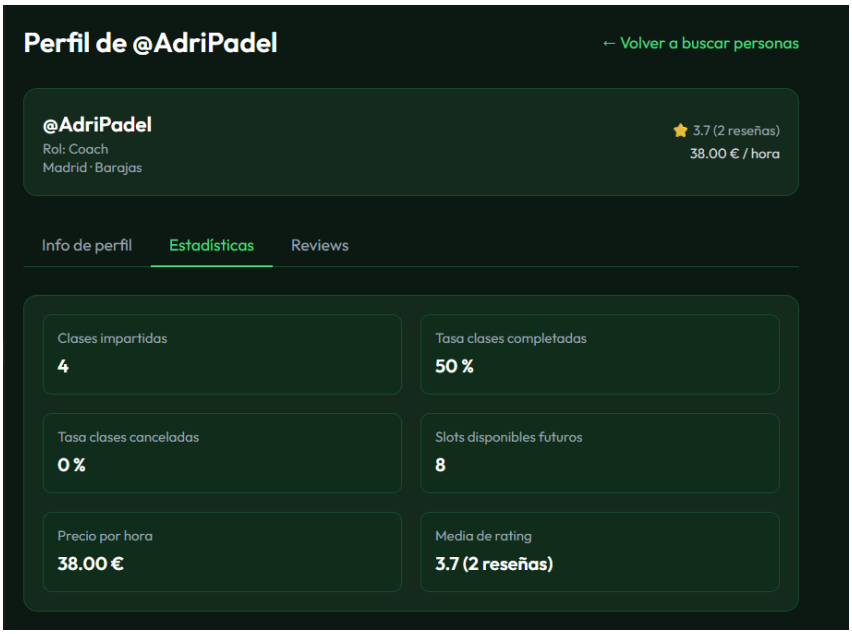

Ilustración 35: Captura de un perfil público con estadísticas y reseñas. (1/2)

Ilustración 36: Captura de un perfil público con estadísticas y reseñas. (2/2)

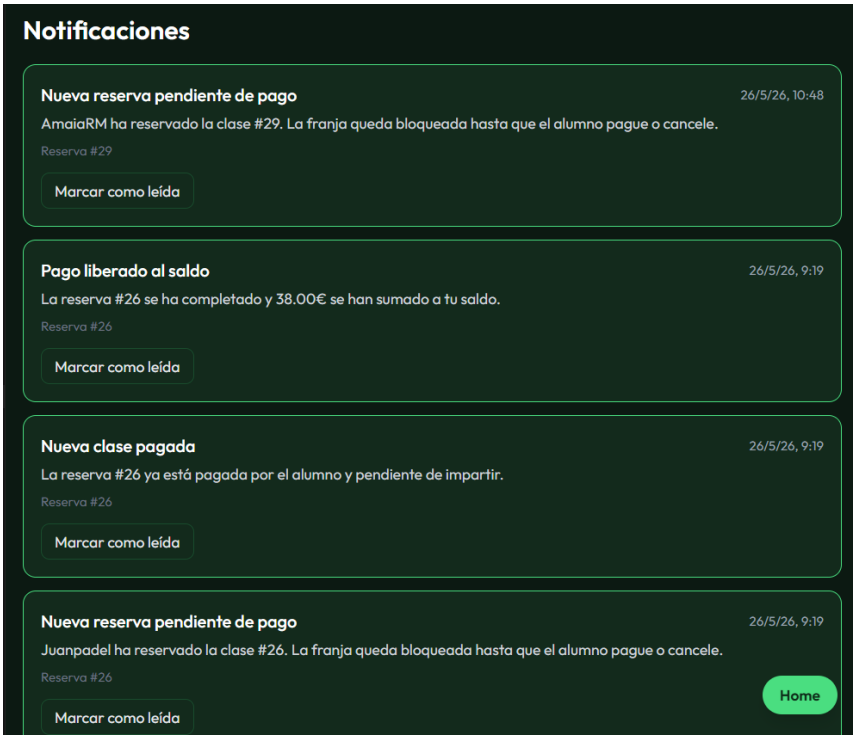

Ilustración 37: Captura de la bandeja de notificaciones.

FUNCIONES DEL ADMINISTRADOR

El administrador es un perfil interno cuya función principal es resolver las incidencias que no se cierran entre las partes, decidiendo entre el reembolso al alumno, la liberación al profesor o un reparto parcial del importe.

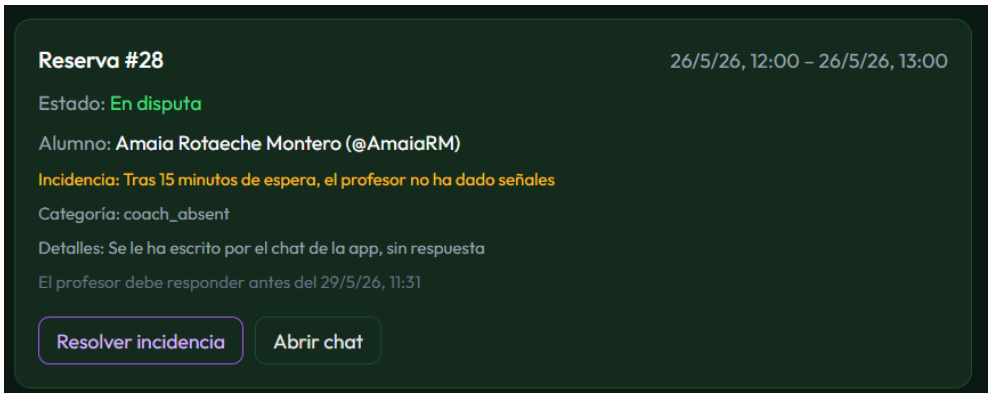

Ilustración 38: Captura de la resolución de una incidencia por parte del administrador. (1/2)

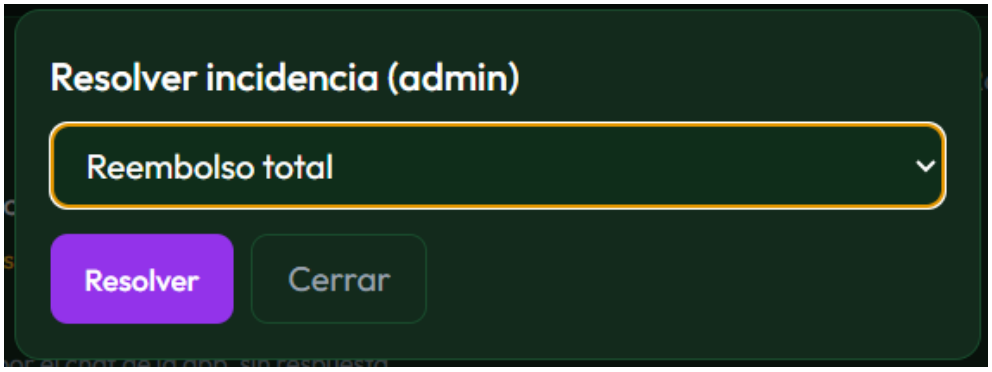

Ilustración 39: Captura de la resolución de una incidencia por parte del administrador. (2/2)

SÍNTESIS

Este recorrido cubre las funciones principales de PlayMatch para cada perfil. Con el manual de usuario completo, el capítulo siguiente detalla cómo instalar y poner en marcha la aplicación.

ANEXO II: MANUAL DE INSTALACIÓN Y PUESTA EN MARCHA

Este capítulo explica cómo instalar y poner en marcha PlayMatch, tanto en un entorno de desarrollo como mediante contenedores. Se asume un conocimiento básico de la línea de comandos.

REQUISITOS PREVIOS

Antes de instalar la aplicación conviene disponer de las herramientas que recoge la *Tabla 30*. El backend requiere Python; el frontend, Node.js; y, de forma opcional, Docker permite un despliegue sin instalar las dependencias en el sistema.

Tabla 30: Requisitos previos para la instalación.

Componente	Requisito
Python	Versión 3.11 o superior (para el backend).
Node.js	Versión 18 o superior (para el frontend).
Git	Para clonar el repositorio del proyecto.
Docker (opcional)	Para el despliegue en contenedores sin instalar dependencias.
Navegador web	Para acceder a la interfaz de la aplicación.

INSTALACIÓN EN ENTORNO DE DESARROLLO

BACKEND

El backend se instala dentro de un entorno virtual de Python. Tras clonar el repositorio, se crea el entorno y se instalan las dependencias; a continuación, se aplican las migraciones para crear la base de datos y se arranca el servidor. El código 6 resume los pasos.

```
# 1. Clonar el repositorio
git clone <url-del-repositorio>
cd playmatch

# 2. Crear y activar el entorno virtual
python -m venv .venv
source .venv/bin/activate        # En Windows: .venv\Scripts\activate

# 3. Instalar dependencias y preparar la base de datos
pip install -r requirements.txt
alembic upgrade head

# 4. Arrancar la API (http://127.0.0.1:8000, documentacion en /docs)
uvicorn app.main:app --reload
```

Código 6: Instalación y arranque del backend en desarrollo.

FRONTEND

El frontend se instala con el gestor de paquetes de Node. Desde el directorio frontend se instalan las dependencias y se arranca el servidor de desarrollo, que se comunica con la API mediante un proxy. El código 7 muestra los comandos.

```
cd frontend
npm install            # solo la primera vez
npm run dev            # interfaz en http://localhost:5173
```

Código 7: Instalación y arranque del frontend en desarrollo.

Con ambos procesos en marcha, la API queda disponible en el puerto 8000 —con su documentación interactiva en la ruta /docs— y la interfaz, en el puerto 5173.

CONFIGURACIÓN MEDIANTE VARIABLES DE ENTORNO

Los parámetros sensibles y dependientes del entorno se definen en un archivo .env situado en la raíz del proyecto, que no debe incluirse en el control de versiones. El código 8 muestra un ejemplo; en producción es imprescindible sustituir la clave de firma por un valor largo y aleatorio. El significado de cada variable se detalló en la *Tabla 23*.

```
DATABASE_URL=sqlite:///./playmatch.db
JWT_SECRET_KEY=una_clave_larga_y_aleatoria
JWT_ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=60
CORS_ORIGINS=
```

Código 8: Ejemplo de archivo .env.

DESPLIEGUE CON DOCKER

Como alternativa al entorno de desarrollo, Docker permite levantar toda la aplicación con un único comando, sin instalar Python ni Node en el sistema. Basta con definir una clave secreta y construir los contenedores; la aplicación queda accesible en el puerto 8080. El código 9 resume el proceso.

```
# Definir una clave secreta robusta (recomendado)
export JWT_SECRET_KEY="una_clave_larga_y_aleatoria"

# Construir y levantar los contenedores
docker compose up --build

# La aplicacion queda disponible en http://localhost:8080
# Para detenerla: Ctrl+C  (o docker compose down)
```

Código 9: Despliegue de la aplicación con Docker Compose.

PUESTA EN MARCHA Y VERIFICACIÓN

Una vez arrancada, conviene comprobar que el servicio responde accediendo a la ruta de estado o a la documentación interactiva de la API. A partir de ahí, se pueden registrar usuarios desde la propia interfaz y empezar a operar. El rol de administrador, al no ofrecerse en el registro público, se asigna manualmente sobre la base de datos.

SÍNTESIS

Con estos pasos, PlayMatch queda instalada y operativa, ya sea para el desarrollo o para una demostración reproducible mediante contenedores. El capítulo siguiente presenta la memoria económica y de planificación del proyecto, antes de las conclusiones finales.

ANEXO III: GLOSARIO DE TÉRMINOS

La *Tabla 31* recoge, por orden alfabético, los principales términos técnicos empleados en la memoria, con una definición breve y accesible.

Tabla 31: Glosario de términos técnicos.

Término	Definición
API REST	Interfaz que permite la comunicación entre programas por HTTP mediante operaciones sobre recursos.
Alembic	Herramienta de migraciones del esquema de base de datos para SQLAlchemy.
Backend	Parte de la aplicación que se ejecuta en el servidor: lógica de negocio y datos.
bcrypt	Algoritmo de hash diseñado para almacenar contraseñas de forma segura.
Contenedor	Unidad aislada que empaqueta una aplicación y sus dependencias (Docker).
CORS	Mecanismo que controla qué orígenes web pueden realizar peticiones a una API.
Docker	Tecnología de contenedores para ejecutar aplicaciones de forma reproducible.
Endpoint	Ruta de una API asociada a una función concreta.
Escrow	Esquema de retención del pago hasta que se cumple la condición acordada.
FastAPI	Framework de Python para construir API REST.
Frontend	Parte de la aplicación que se ejecuta en el navegador: la interfaz de usuario.
Hash	Resultado irreversible de aplicar una función criptográfica a un dato.
Haversine	Fórmula que calcula la distancia entre dos puntos sobre la superficie terrestre.
Hook	Función de React que añade lógica reutilizable a los componentes.
JSON	Formato de texto ligero para intercambiar datos estructurados.
JWT	Token firmado que transporta la información de la sesión (JSON Web Token).
Matching	Mecanismo de recomendación que empareja alumnos y profesores.
Middleware	Componente que intercepta las peticiones para aplicar lógica común.
MVP	Producto mínimo viable: primera versión funcional con lo esencial.
Nginx	Servidor web ligero, usado también como proxy inverso.
Nominatim	Servicio de geocodificación basado en OpenStreetMap.
OAuth2	Estándar de autorización; en este proyecto, el flujo de tipo password.
ORM	Mapeador objeto-relacional que traduce clases a tablas (SQLAlchemy).
Proxy	Componente que reenvía peticiones de un punto a otro.
Pydantic	Biblioteca de validación de datos por tipos en Python.
SPA	Aplicación de página única, que se actualiza sin recargar la página.
SQLite	Base de datos relacional ligera basada en un único archivo.
Token	Credencial que acredita una sesión autenticada.
Uvicorn	Servidor que ejecuta aplicaciones web asíncronas de Python (ASGI).

ANEXO IV: DICCIONARIO DE DATOS

Este anexo detalla los campos principales de cada entidad del modelo de datos descrito en el capítulo 5.6.5. Por brevedad, se omiten algunos campos auxiliares y de marca temporal.

ENTIDAD USER

Cuenta de usuario.

Tabla 32: Campos de la entidad User.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador único del usuario.
email	Texto, único	Correo electrónico, usado también para iniciar sesión.
username	Texto, único	Nombre de usuario público.
hashed_password	Texto	Contraseña cifrada con bcrypt.
role	Enumerado	Rol del usuario: student, coach o admin.
created_at	Fecha y hora	Fecha de creación de la cuenta.

ENTIDAD PROFILE

Perfil asociado a un usuario (1:1).

Tabla 33: Campos de la entidad Profile.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador del perfil.
user_id	Entero (FK)	Usuario propietario del perfil.
full_name	Texto	Nombre completo.
city / district	Texto	Ciudad y distrito seleccionados.
lat / lon	Decimal	Coordenadas aproximadas para el matching.
level	Entero (1–5)	Nivel de juego.
price_per_hour	Decimal	Tarifa (profesor) o presupuesto máximo (alumno).
coach_title_name / description	Texto	Titulación y descripción del profesor.

ENTIDAD AVAILABILITYSLOT

Franja de disponibilidad de un profesor.

Tabla 34: Campos de la entidad AvailabilitySlot.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador de la franja.
coach_id	Entero (FK)	Profesor que la publica.
start_time / end_time	Fecha y hora	Inicio y fin de la franja.

ENTIDAD BOOKING

Reserva de una clase.

Tabla 35: Campos de la entidad Booking.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador de la reserva.
student_id / coach_id	Entero (FK)	Alumno y profesor de la reserva.
availability_slot_id	Entero (FK)	Franja reservada.
start_time / end_time	Fecha y hora	Horario de la clase.
status	Enumerado	Estado: pending, confirmed, completed, cancelled o disputed.
cancelled_by / refund_percent	Texto / Decimal	Datos de la cancelación.
dispute_*	Varios	Campos de la incidencia (motivo, plazos, respuesta, resolución).

ENTIDAD PAYMENT

Pago asociado a una reserva.

Tabla 36: Campos de la entidad Payment.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador del pago.
booking_id	Entero (FK)	Reserva a la que corresponde.
status	Texto	Estado: held, released, refunded, partial o failed.
reference	Texto	Referencia del pago (simulado).
amount_total	Decimal	Importe total.
amount_refunded / amount_released	Decimal	Importes reembolsado y liberado.

ENTIDAD PAYMENTMETHOD

Tarjeta guardada en el monedero.

Tabla 37: Campos de la entidad PaymentMethod.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador del método de pago.
user_id	Entero (FK)	Usuario propietario.
brand	Texto	Marca de la tarjeta.
first4 / last4	Texto	Cuatro primeros y últimos dígitos (datos parciales).
exp_month / exp_year	Entero	Mes y año de caducidad.

ENTIDAD CHATMESSAGE

Mensaje del chat de una reserva.

Tabla 38: Campos de la entidad ChatMessage.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador del mensaje.
booking_id	Entero (FK)	Reserva a la que pertenece.
sender_id	Entero (FK)	Usuario que lo envía.
content	Texto	Contenido del mensaje.
created_at	Fecha y hora	Fecha de envío.

ENTIDAD CHATREADSTATE

Estado de lectura del chat por usuario.

Tabla 39: Campos de la entidad ChatReadState.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador del registro.
booking_id / user_id	Entero (FK)	Reserva y usuario.
last_read_at	Fecha y hora	Última lectura del chat.

ENTIDAD REVIEW

Valoración de una clase completada.

Tabla 40: Campos de la entidad Review.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador de la valoración.
booking_id	Entero (FK), único	Reserva valorada.
reviewer_id / reviewee_id	Entero (FK)	Autor y profesor valorado.
rating / overall_score	Entero / Decimal	Valoración global agregada.
*_score (9 ejes)	Entero (1–5)	Puntuación de cada dimensión.
comment	Texto	Comentario opcional.

ENTIDAD NOTIFICATION

Aviso dirigido a un usuario.

Tabla 41: Campos de la entidad Notification.

Campo	Tipo	Descripción
id	Entero (PK)	Identificador del aviso.
user_id	Entero (FK)	Destinatario.
booking_id	Entero (FK), opcional	Reserva relacionada, si aplica.
title / message	Texto	Título y cuerpo del aviso.
level	Texto	Tipo de aviso (info, warning, etc.).
read_at	Fecha y hora	Momento de lectura (nulo si no leída).

ANEXO V: LISTA DE COMPROBACIÓN DE SEGURIDAD PREVIA AL DESPLIEGUE

Antes de poner el sistema en producción conviene revisar las comprobaciones recogidas en la *Tabla 42*, que sintetizan las buenas prácticas descritas en el capítulo 5.4.7.

Tabla 42: Lista de comprobación de seguridad previa al despliegue.

Comprobación	**Detalle**
Clave de firma robusta	Sustituir JWT_SECRET_KEY por un valor largo y aleatorio.
Archivo .env protegido	Mantener el archivo .env fuera del control de versiones.
HTTPS habilitado	Servir la aplicación siempre sobre HTTPS en producción.
Base de datos no versionada	No incluir el archivo de base de datos en el repositorio.
CORS restringido	Limitar los orígenes permitidos a los dominios de producción.
Errores sin detalle	Ocultar los mensajes internos de error al usuario en producción.
Contraseñas cifradas	Verificar que se almacenan con bcrypt, nunca en claro.
Datos de pago minimizados	Confirmar que no se guardan el número completo ni el código de seguridad.
Separación de entornos	Mantener configuraciones distintas para desarrollo y producción.
Copias de seguridad	Establecer copias de seguridad periódicas de la base de datos.

ANEXO VI: DOCUMENTACIÓN INTERACTIVA DE LA API

FastAPI genera automáticamente una documentación interactiva de la API (Swagger UI), accesible en la ruta /docs, que permite explorar y probar los endpoints. Se sugiere incluir una captura de dicha documentación.

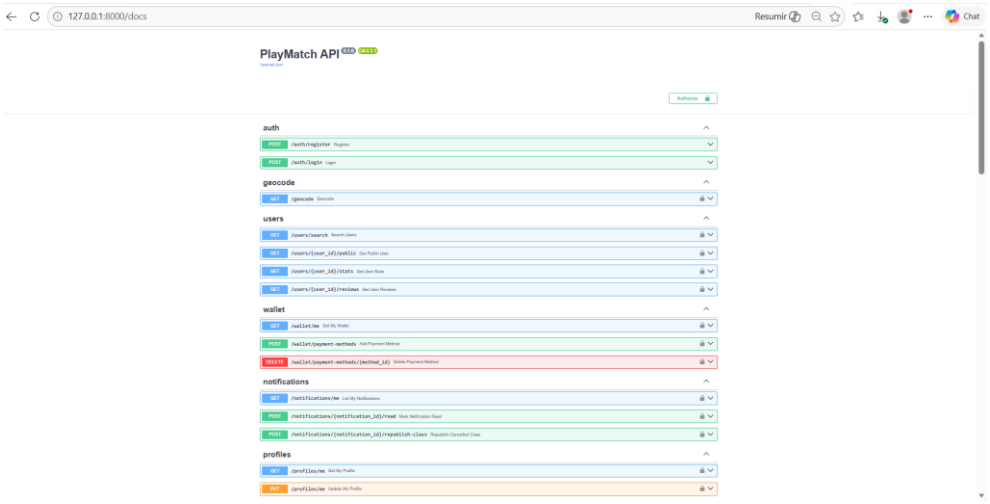

Ilustración 40: Captura de la documentación interactiva de la API (Swagger UI) en la ruta /docs. (1/2)

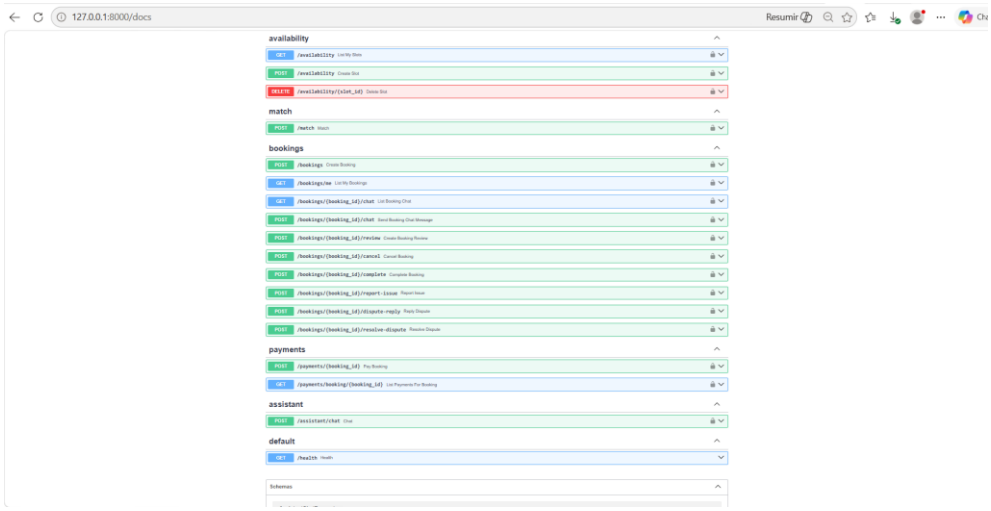

Ilustración 41: Captura de la documentación interactiva de la API (Swagger UI) en la ruta /docs. (2/2)

ANEXO VII: ALINEAMIENTO CON LOS ODS (OBJETIVOS DE DESARROLLO SOSTENIBLE)

El proyecto PlayMatch se alinea con varios de los Objetivos de Desarrollo Sostenible (ODS) establecidos por la ONU en la Agenda 2030. La Tabla 43 recoge los ODS más directamente relacionados con el proyecto y justifica la aportación de la plataforma a cada uno de ellos.

Tabla 43: Alineamiento de Playmatch con los objetivos de desarrollo sostenible

ODS	Nombre del objetivo	Aportación de PlayMatch
ODS 3	Salud y bienestar	PlayMatch facilita el acceso a clases de pádel de calidad, reduciendo las barreras para encontrar entrenadores cualificados. Al profesionalizar y digitalizar la relación alumno-entrenador, la plataforma contribuye a que más personas puedan practicar deporte de forma regular y con guía técnica adecuada, impactando positivamente en su salud y bienestar.
ODS 8	Trabajo decente y crecimiento económico	La plataforma genera nuevas oportunidades para entrenadores autónomos, mejorando su visibilidad, la gestión de su agenda y la regularidad de sus ingresos. Al estructurar la relación económica con el pago en retención (escrow), se protege la remuneración del profesional y se fomenta el autoempleo en la economía local.
ODS 9	Industria, innovación e infraestructura	El diseño e implementación de PlayMatch supone el desarrollo de una infraestructura digital innovadora que digitaliza un sector tradicionalmente informal. El algoritmo de matching, la arquitectura en capas y el sistema de pagos constituyen aportaciones técnicas que contribuyen a la innovación tecnológica en el ámbito de los servicios deportivos.
ODS 10	Reducción de las desigualdades	Al centralizar la oferta de entrenadores y hacerla accesible mediante criterios objetivos (nivel, ubicación, precio), PlayMatch reduce la dependencia de las redes informales de contactos. Esto democratiza el acceso a la formación deportiva, permitiendo que cualquier jugador, independientemente de sus relaciones previas, pueda encontrar un entrenador adecuado.

ODS 17	Alianzas para lograr los objetivos	La arquitectura de PlayMatch como plataforma de intermediación multilateral fomenta la colaboración entre jugadores, entrenadores y, potencialmente, clubes e instituciones deportivas. Su modelo abierto y su API REST están diseñados para facilitar integraciones con terceros y construir un ecosistema digital compartido.

ANEXO VIII: DECLARACIÓN DE ORIGINALIDAD

Declaro bajo mi responsabilidad que el Proyecto presentado con el título **PlayMatch: diseño e implementación de una aplicación web para la conexión entre profesores y alumnos de pádel**e la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico **2025/26** es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial[2]

Declaro bajo mi responsabilidad que (indicar la opción correcta):

☐ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

X He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): "*La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente*". En concreto, las Inteligencia Artificial ha sido empleada para:

He utilizado Inteligencia Artificial en la elaboración del presente documento siempre dentro de las condiciones permitidas por la Universidad Pontificia Comillas, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024). En concreto, la IA ha sido empleada para las siguientes actividades previas o de apoyo a la tarea: • **Brainstorming y estructuración inicial:** explorar posibles enfoques del proyecto, esbozar la organización del repositorio de trabajo y de la memoria y generar ideas sobre funcionalidades del sistema antes de desarrollarlas de forma independiente.

[2] Esta declaración se refiere al uso de la Inteligencia Artificial generativa para realizar los documentos del Proyecto (Anexo B y Memoria). No aplica a Proyectos donde, por su naturaleza, deban emplear inteligencia artificial como parte de los mismos (aplicación de técnicas de aprendizaje automático, redes neuronales, análisis de datos…)

- **Identificación de referencias preliminares:** localizar fuentes relevantes sobre tecnologías, marcos regulatorios y trabajos relacionados, usada conjuntamente con otras herramientas y contrastando y validando manualmente los resultados.
- **Síntesis de conceptos técnicos:** comprender y resumir protocolos y herramientas específicas —como OAuth2, JWT o Docker Compose— antes de redactar su descripción en el documento.
- **Corrección puntual de errores de código:** La asistencia integrada de Visual Studio, “Copilot”, para la resolución de errores puntuales, especialmente relacionados con la lógica de "ubicación" en el algoritmo de matching y el sistema de pagos con retención.
- **Revisión y mejora del estilo:** pulir la redacción de determinados párrafos, manteniendo en todo momento el contenido técnico y las decisiones de diseño como responsabilidad exclusiva del autor.
- **Traducción y adaptación de textos:** trasladar fragmentos del inglés al español en el proceso de documentación.

Firmado (alumno): Amaia Rotaeche Montero
Fecha: 04/06/2026