Octobets

David Mezcua Diago

CFGS Desarrollo de Aplicaciones Web

Curso 2025

Tutor Individual: Lorenzo José Díaz de Haro

RESUMEN

Octobets nace como una plataforma web que traslada al entorno digital las apuestas informales de toda la vida: las que se hacen entre amigos, compañeros o comunidades locales y que rara vez cuentan con un sistema claro para llevar su control. El resultado es un espacio donde cualquier usuario puede proponer un reto, fijar sus propias reglas y dejar que los demás participen apostando fichas virtuales. Todo ello acompañado de registros históricos, autenticación segura con JWT y pagos simulados mediante Stripe. Desde la primera visita el portal presenta las apuestas públicas, y aunque cualquiera puede curiosear, tan solo los registrados tienen acceso a la creación y a la participación real. Con ello se consigue abrir la puerta a la curiosidad del visitante sin descuidar la seguridad ni la trazabilidad de los movimientos.

Para dar vida a la idea se combinan Spring Boot, Angular 19 y Tailwind CSS en una arquitectura por capas que separa presentación, lógica y datos, alojada sobre un VPS contenedorizado con Docker y desplegada de forma continua gracias a GitHub Actions. El proyecto se ha desarrollado en solitario bajo un Scrum “de un solo miembro”, lo que ha permitido ajustar el alcance —dejando para futuras versiones el módulo de juegos de casino— sin sacrificar calidad ni fechas. El resultado es un producto sólido, ligero y, sobre todo, preparado para crecer: bastará añadir la lógica de cierre automático, notificaciones en tiempo real o nuevas pasarelas de pago para que la plataforma dé el siguiente salto.

Índice

1 Introducción

1.1 Presentación del proyecto

El propósito fundamental de este proyecto es la creación y desarrollo de **Octobets**, una plataforma web centrada en la generación de apuestas personalizadas entre usuarios. A diferencia de las casas de apuestas tradicionales, donde solo se puede apostar sobre eventos oficiales como partidos de primera división o competiciones profesionales, Octobets permite que sean los propios usuarios quienes creen y gestionen sus propias apuestas, adaptadas a su entorno y realidad.

Esto abre la puerta a apostar, por ejemplo, al resultado de un partido de fútbol entre amigos, a quién ganará un torneo local o a cualquier otro evento cotidiano, siempre dentro de un entorno informal.

En su concepción inicial, el proyecto contemplaba también una sección dedicada a juegos de casino, como máquinas tragaperras (*slots*). Sin embargo, tras un análisis de viabilidad técnica, legal y temporal realizado durante las primeras fases del desarrollo, se decidió dejar fuera esta funcionalidad y centrar el alcance del proyecto exclusivamente en el sistema de apuestas.

Entre las funcionalidades implementadas destacan: la creación de apuestas por parte de los usuarios, la participación en ellas mediante un sistema de fichas virtuales, la resolución de estas, y la integración de pagos reales mediante Stripe para la compra de fichas. Todos los datos relevantes —usuarios, apuestas, resultados y transacciones— se gestionan de forma persistente en una base de datos MySQL, asegurando una estructura sólida y coherente para el funcionamiento de la plataforma.

1.2 Justificación y contexto en el que se desarrolla el Proyecto Integrado

En la actualidad, apostar de manera informal entre amigos o grupos cercanos sobre eventos cotidianos es una práctica habitual que despierta un interés creciente, especialmente entre personas jóvenes o dentro de comunidades lo-

cales. Sin embargo, hasta ahora, no existe una solución tecnológica extendida o accesible que permita gestionar este tipo de apuestas de manera sencilla, segura y cómoda para los usuarios. Habitualmente, este tipo de apuestas informales se limitan a anotaciones en papel o en chats grupales, lo que dificulta tanto el seguimiento como la gestión eficiente y transparente de los resultados.

En este contexto surge **Octobets**, no como un competidor de las casas de apuestas tradicionales, sino como una herramienta complementaria orientada a un público diferente. El objetivo principal no es apostar en competiciones oficiales o eventos profesionales, sino proporcionar una plataforma específica para gestionar apuestas personalizadas en contextos informales, permitiendo que esta actividad, que habitualmente es difícil de llevar a cabo de manera organizada, sea mucho más sencilla y atractiva para los usuarios.

En las primeras fases de diseño se contempló crear una interfaz especialmente interactiva y llamativa visualmente, planteando el uso de librerías modernas como GSAP. Sin embargo, debido a limitaciones de tiempo, esta propuesta no llegó finalmente a implementarse.

En la parte del *backend*, se utilizó Spring Boot, asegurado mediante JWT y Spring Security, para conseguir un sistema robusto y escalable capaz de gestionar usuarios, apuestas y transacciones de forma segura. El *frontend* fue desarrollado con Angular y Tailwind CSS, facilitando así una interfaz amigable, modular y adaptable a diferentes pantallas. Además, toda la información generada por la aplicación (usuarios, apuestas y transacciones) se gestiona de forma persistente con MySQL, escogido por su eficiencia y fiabilidad.

Durante el desarrollo, se realizó un análisis detallado que puso de manifiesto que incluir la sección de juegos de casino (*slots*), inicialmente prevista, exigía una cantidad de tiempo y esfuerzo considerablemente superior a la disponible, especialmente debido a las estrictas exigencias regulatorias asociadas a los juegos de azar. Estos requisitos habrían implicado una inversión elevada en términos de tiempo, documentación y recursos adicionales. Ante esta situación, se optó por ajustar el alcance del proyecto concentrando los esfuerzos en la funcionalidad completa y la calidad de la sección de apuestas personalizadas, asegurando así la entrega de un producto viable dentro de los plazos

establecidos.

1.3 Objetivos del Proyecto Integrado

Objetivo general

- Construir una plataforma web funcional y eficiente capaz de gestionar un gran volumen de datos, centrada en la creación y resolución de apuestas personalizadas y en el seguimiento financiero de los usuarios (movimientos de fichas, ingresos y retiros).

Objetivos específicos del sistema

1. **Gestión de apuestas personalizadas**
    - Permitir a cualquier usuario crear, publicar y cerrar apuestas propias con reglas definidas *ad hoc*.
    - Registrar de forma persistente cada apuesta y sus resultados, garantizando trazabilidad histórica.
2. **Seguimiento financiero y monedero virtual**
    - Implementar un sistema de fichas respaldado por pagos reales mediante Stripe, con depósitos, ganancias y retiros auditables.
    - Mostrar al usuario su historial económico completo (ingresos, apuestas realizadas, beneficios y pérdidas).
3. **Seguridad y autenticación**
    - Proteger todas las rutas críticas mediante JWT y Spring Security, controlando roles y permisos.
    - Cifrar contraseñas y aplicar buenas prácticas en el backend.
4. **API REST modular y escalable**
    - Diseñar endpoints REST claros y versionables.

   - Separar lógica de dominio (apuestas), infraestructura (pagos) y presentación para facilitar futuras extensiones (ej. módulo de juegos).

5. **Front-end intuitivo y rápido**
   - Desarrollar la interfaz con Angular y Tailwind CSS, priorizando tiempos de carga bajos y navegación fluida.
6. **Despliegue y operaciones**
   - Automatizar el pipeline CI/CD en GitHub Actions con empaquetado de Docker y despliegue en VPS.
7. **Pruebas y calidad**
   - Cubrir la lógica crítica con test unitarios.
   - Realizar pruebas de carga para validar que la plataforma soporta picos de usuarios y apuestas simultáneas.

Objetivos de calidad y rendimiento

- **Modularidad y mantenibilidad:** seguir una estructura de paquetes limpia (model.dto, repository.dao, service, rest.controller), aplicar principios SOLID y preparar la base para futuras funcionalidades.
- **Escalabilidad horizontal:** dejar preparada la configuración para escalar instancias del backend y la base de datos si la demanda crece.

Objetivos de aprendizaje para el alumno

- Consolidar competencias *full-stack* (Spring Boot + Angular) aplicando seguridad, persistencia y UI moderna en un caso real.
- Aprender a integrar pasarelas de pago (Stripe) y gestionar flujos de dinero virtual de forma segura.

- Desarrollar y desplegar un proyecto contenedorizado con CI/CD autónomo.
- Aplicar metodologías ágiles y gestionar el *backlog* priorizando funcionalidades viables dentro de los plazos.

1.4 Metodología y gestión del cambio de alcance

Para el desarrollo de **Octobets** se adoptó una metodología ágil basada en *Scrum*, adaptada a un proyecto llevado a cabo por una sola persona. Se optó por este enfoque por su capacidad para facilitar un desarrollo rápido, iterativo y flexible, permitiendo avanzar por fases funcionales claras y adaptarse fácilmente a los cambios que fueron surgiendo durante el proceso.

La gestión del cambio de alcance se llevó a cabo de la siguiente forma:

- En una revisión intermedia, se valoró que implementar el módulo de juegos de casino suponía un esfuerzo muy superior al previsto inicialmente.
- A esto se sumaba la complejidad técnica/legal del desarrollo de juegos de azar, que habría requerido una inversión de tiempo y recursos que excedía el margen del proyecto.
- Se decidió centrar el alcance exclusivamente en la parte de apuestas personalizadas, priorizando así el núcleo funcional del sistema y asegurando su entrega con garantías de calidad.
- Aunque no se desarrolló la parte de los juegos, se dejó planteada la estructura general junto al sistema para poder añadir este módulo en un futuro sin necesidad de rehacer elementos clave.
- Esta decisión permitió enfocar todos los esfuerzos en completar correctamente las funcionalidades principales: creación de apuestas, participación con fichas virtuales, integración de pagos y sistema de resolución.

Gracias al enfoque iterativo de *Scrum*, el proyecto pudo evolucionar con agilidad, priorizando lo realmente viable y ajustando la planificación sin comprometer el resultado final.

1.5 Planificación del proyecto

A continuación se expone la planificación del proyecto, dividida en fases cronológicas. Primero se presenta el cronograma original previsto al inicio del desarrollo, seguido del cronograma real, que refleja las adaptaciones llevadas a cabo tras el reajuste del alcance.

Cronograma original

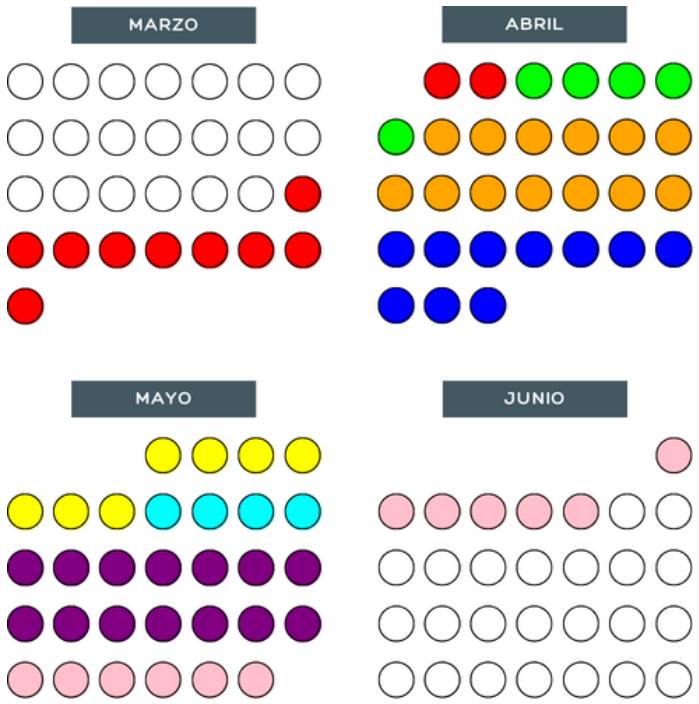

Figura 1: Cronograma original del proyecto

Fase 1 (Rojo) – 24/03 al 02/04

- Investigación de tecnologías viables para el proyecto.
- Diseño inicial de la interfaz de usuario.

- Definición de las funcionalidades principales: sistema de apuestas y juegos de casino.

Fase 2 (Verde) – 03/04 al 07/04

- Configuración del backend con Spring Boot.
- Desarrollo de los endpoints principales y primeras pruebas de persistencia.
- Implementación del modelo de base de datos y su mapeo.

Fase 3 (Naranja) – 08/04 al 20/04

- Desarrollo del frontend con Angular.
- Implementación de la lógica de apuestas personalizadas.
- Comienzo de la estructura base para los juegos de casino.

Fase 4 (Azul) – 21/04 al 30/04

- Desarrollo de animaciones para los juegos con la librería GSAP.

Fase 5 (Amarillo) – 01/05 al 07/05

- Integración del sistema de pagos con Stripe.
- Aplicación de JWT para la autenticación y autorización segura.

Fase 6 (Cian) – 07/05 al 11/06

- Despliegue de la aplicación en servidor.

Fase 7 (Morado) – hasta el 25/05

- Pruebas finales y corrección de errores.

Fase 8 (Rosa) – hasta el 06/06

- Elaboración de la presentación para la defensa.
- Redacción final de la memoria.

Cronograma real

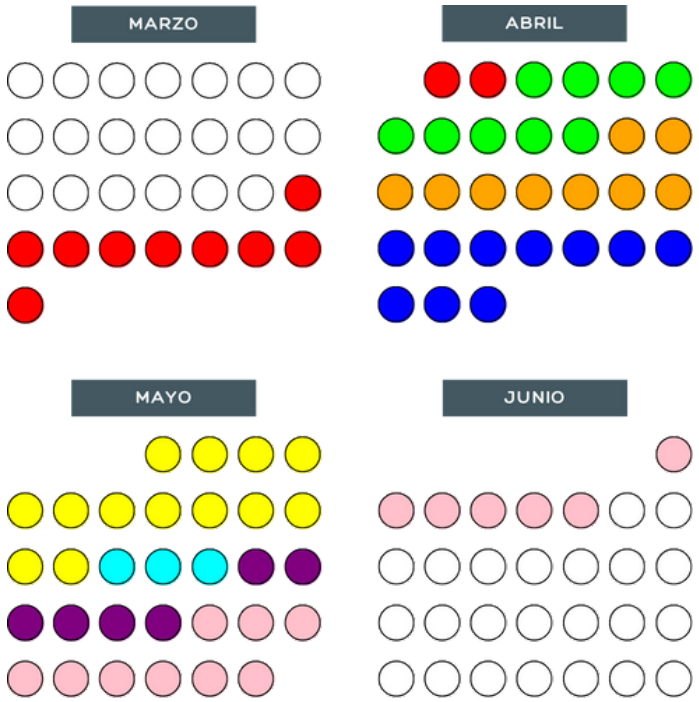

Figura 2: Cronograma real del proyecto tras el cambio de alcance

Fase 1 (Rojo) – 24/03 al 02/04

- Igual que en la planificación inicial: investigación, primeros diseños y definición del proyecto con alcance completo (apuestas + juegos de casino).

Fase 2 (Verde) – 02/04 al 11/04

- Desarrollo del backend con Spring Boot.
- Creación de endpoints principales.
- Diseño e implementación del modelo de base de datos.

Fase 3 (Naranja) – 12/04 al 20/04

- Reevaluación del alcance del proyecto.
- Se analiza el coste en tiempo y recursos legales que implicaría implementar juegos de casino.
- Se decide limitar el desarrollo a la parte de apuestas personalizadas, reorganizando tareas en función del nuevo enfoque.

Fase 4 (Azul) – 21/04 al 30/04

- Desarrollo del frontend centrado exclusivamente en apuestas.
- Comienza la implementación real de la lógica visual y funcional bajo el nuevo planteamiento.

Fase 5 (Amarillo) – 01/05 al 13/05

- Integración del sistema de pagos mediante Stripe.
- Aplicación de seguridad con JWT y Spring Security.

Fase 6 (Cian) – 14/05 al 16/05

- Despliegue de la aplicación mediante Docker y GitHub Actions en un VPS.

Fase 7 (Morado) – 16/05 al 21/05

- Pruebas finales del sistema completo.
- Resolución de errores detectados y mejoras de estabilidad.

Fase 8 (Rosa) – 22/05 al 06/06

- Preparación de la presentación para la defensa del proyecto.
- Redacción de la memoria técnica y documentación complementaria.

1.6 Recursos del proyecto

El desarrollo de **Octobets** ha requerido distintos recursos de hardware y software, todos ellos accesibles y adecuados para llevar a cabo un proyecto de tipo *full-stack* en un entorno académico.

Recursos de hardware

- **Equipos de desarrollo:** Se han utilizado dos ordenadores personales (un portátil y un PC de sobremesa), ambos con sistema operativo Windows 11, para las tareas de programación, pruebas, diseño de interfaz y documentación.
- **Servidor de despliegue (VPS):** La aplicación final se ha desplegado en un servidor VPS, donde se alojan el *backend*, el *frontend* y la base de datos, permitiendo su acceso desde cualquier dispositivo con conexión a internet.

Recursos de software

- **Sistemas operativos:**
  - Windows 11
- **Backend:**
  - Spring Boot
  - Java (JDK 21)
  - Spring Security y JWT para la autenticación
  - Maven para la gestión de dependencias
- **Frontend:**
  - Angular
  - Tailwind CSS como sistema de estilos

    - Librería GSAP (planeada inicialmente para animaciones avanzadas, pero finalmente no implementada)

- **Base de datos:**
    - MySQL
    - Cliente de gestión: MySQL Workbench
- **Diseño de la interfaz de usuario:**
    - Figma, para la elaboración de prototipos visuales y diagramas.
    - Illustrator, para el diseño de ilustraciones
- **Control de versiones:**
    - Git, utilizando GitHub.
- **Cliente para pruebas de API REST:**
    - Postman
- **Entorno de desarrollo:**
    - Visual Studio Code
- **Node.js y npm:**
    - Utilizados para la gestión del entorno del frontend (Angular)
- **Despliegue y automatización:**
    - Docker, para la contenerización de la aplicación
    - GitHub Actions, como herramienta de integración y despliegue continuo
- **Navegadores usados en pruebas:**
    - Google Chrome
    - Mozilla Firefox

2 Análisis de requisitos

2.1 Descripción de actores

A continuación se describen los actores principales identificados dentro de la plataforma **Octobets**, detallando sus roles y funciones específicas en el sistema:

Usuario Registrado (Apostador): Es cualquier usuario que ha completado el proceso de registro e inicio de sesión en la plataforma. Este tipo de usuario puede crear sus propias apuestas personalizadas, participar en las apuestas creadas por otros usuarios y realizar ingresos de fichas a través del sistema de pagos integrado con Stripe.

Usuario No Registrado (Visitante): Representa a cualquier persona que accede a la plataforma sin estar registrada o sin haber iniciado sesión. Este actor tiene acceso exclusivamente a la visualización de las apuestas existentes, sin poder participar activamente ni crear nuevas apuestas.

Administrador (Admin): Usuario con permisos especiales en la plataforma, cuya función principal es eliminar apuestas en caso de ser necesario. El administrador dispone de un acceso específico desde la interfaz de usuario para realizar esta acción.

Stripe (Pasarela de pagos): Actor externo encargado de gestionar las transacciones económicas reales. Es utilizado por los usuarios registrados para comprar fichas virtuales mediante pagos seguros, facilitando así la operativa financiera en la plataforma.

2.2 Características del sistema

Octobets es una plataforma web centrada en la creación y participación en apuestas personalizadas. Las principales funcionalidades implementadas se agrupan en los siguientes módulos:

Autenticación y gestión de usuarios: Los usuarios pueden registrarse, iniciar sesión y cerrar sesión de forma segura. El sistema aplica autenticación basada en tokens JWT, y está protegido mediante Spring Security para garantizar la privacidad y la integridad de las sesiones.

Creación y participación en apuestas personalizadas: Cualquier usuario registrado puede crear una apuesta, configurando manualmente sus opciones. Otros usuarios registrados pueden participar en apuestas abiertas, seleccionando una opción y apostando fichas virtuales. Todas las apuestas son visibles públicamente, incluso para usuarios no registrados (modo solo lectura).

Visualización pública de apuestas: La plataforma permite a cualquier visitante, sin necesidad de iniciar sesión, navegar por el listado de apuestas existentes. Esta funcionalidad facilita la exploración general del sistema y permite a los usuarios potenciales familiarizarse con su funcionamiento antes de registrarse.

Sistema de fichas y pagos integrados: Las apuestas se realizan utilizando fichas virtuales. Los usuarios pueden adquirir fichas mediante pagos reales a través de Stripe, utilizando un sistema integrado en la plataforma. El sistema registra todas las transacciones de forma persistente, asociadas al usuario correspondiente.

Resolución de apuestas: Las apuestas no se resuelven automáticamente. La resolución debe realizarla manualmente el creador de la apuesta. El sistema

calcula automáticamente el reparto de fichas una vez se ha declarado la opción ganadora.

Panel administrativo básico: Los usuarios con rol de administrador pueden eliminar apuestas desde la interfaz, permitiendo moderar contenidos en caso de abuso o error. No se ha implementado un panel completo de gestión de usuarios o estadísticas globales.

Historial de movimientos y estado de cartera: Cada usuario puede consultar el saldo actual de fichas y ver un resumen de sus movimientos económicos: ingresos, apuestas realizadas y ganancias. Esta información se muestra desde su perfil una vez iniciada la sesión.

2.3 Especificación de requerimientos funcionales

A continuación, se detallan los requerimientos funcionales identificados en **Octobets**, agrupados por área funcional:

RF.1 – Gestión de usuarios y autenticación

- **RF.1.1** – El sistema debe permitir el registro de nuevos usuarios mediante un formulario que solicite, como mínimo, nombre de usuario, correo electrónico y contraseña.
- **RF.1.2** – Los usuarios registrados deben poder iniciar sesión utilizando sus credenciales.
- **RF.1.3** – El sistema debe mantener la sesión activa mediante autenticación con tokens JWT hasta que el usuario cierre sesión o el token expire.

RF.2 – Creación y visualización de apuestas

- **RF.2.1** – Los usuarios autenticados deben poder crear nuevas apuestas personalizadas indicando título, descripción, opciones disponibles y tipo de apuesta.
- **RF.2.2** – Las apuestas creadas deben estar visibles públicamente para todos los usuarios, incluidos los no registrados.
- **RF.2.3** – El sistema debe permitir consultar el listado de apuestas existentes, filtrando por tipo.
- **RF.2.4** – Cada apuesta debe contar con una vista de detalle accesible desde el listado general.

RF.3 – Participación en apuestas

- **RF.3.1** – Los usuarios registrados deben poder participar en apuestas activas seleccionando una de las opciones disponibles e indicando la cantidad de fichas a apostar.
- **RF.3.2** – El sistema debe bloquear la participación en apuestas cerradas, eliminadas o resueltas.
- **RF.3.3** – Una vez realizada la apuesta, debe quedar registrada en la base de datos asociada al usuario y a la apuesta correspondiente.

RF.4 – Resolución de apuestas

- **RF.4.1** – El creador de una apuesta debe poder marcar la opción ganadora una vez finalizado el evento.
- **RF.4.2** – El sistema debe calcular y repartir automáticamente las fichas entre los usuarios ganadores en proporción a sus apuestas.

RF.5 – Sistema de fichas y pagos

- **RF.5.1** – El sistema debe permitir que los usuarios registrados adquieran fichas mediante pagos reales a través de Stripe.
- **RF.5.2** – Las transacciones deben quedar registradas y asociadas al usuario correspondiente.
- **RF.5.3** – El sistema debe reflejar en tiempo real el saldo de fichas actual del usuario tras cada operación.

RF.6 – Administración básica

- **RF.6.1** – El sistema debe ofrecer a los administradores la posibilidad de eliminar apuestas desde una vista dedicada.
- **RF.6.2** – Las apuestas eliminadas deben desaparecer del listado general y no estar accesibles desde la interfaz pública.

RF.7 – Historial y seguimiento económico

- **RF.7.1** – Cada usuario debe poder consultar el saldo actual de fichas.
- **RF.7.2** – El sistema debe mostrar un resumen de movimientos recientes (apuestas realizadas, ingresos, ganancias).
- **RF.7.3** – Las operaciones deben almacenarse de forma persistente en la base de datos.

2.4 Requerimientos no funcionales del sistema

Además de las funcionalidades principales descritas anteriormente, **Octobets** debe cumplir una serie de requisitos no funcionales que garantizan la calidad, seguridad y mantenibilidad del sistema.

RNF.1 – Seguridad

- **RNF.1.1** – El acceso a funcionalidades restringidas (crear, apostar, resolver) debe estar protegido mediante autenticación JWT.
- **RNF.1.2** – Las contraseñas de los usuarios deben almacenarse de forma cifrada.
- **RNF.1.3** – El sistema debe validar los datos de entrada para prevenir ataques comunes como inyecciones SQL o XSS.

RNF.2 – Persistencia de datos

- **RNF.2.1** – Toda la información relacionada con usuarios, apuestas, participaciones, resultados y movimientos económicos debe almacenarse de forma persistente en una base de datos MySQL.
- **RNF.2.2** – Los datos deben estar estructurados de forma que puedan ser recuperados y consultados eficientemente desde el backend.

RNF.3 – Usabilidad

- **RNF.3.1** – La interfaz debe ser intuitiva, clara y coherente, facilitando la navegación tanto para usuarios nuevos como habituales.
- **RNF.3.2** – La información importante (saldo, estado de apuesta, opciones disponibles) debe mostrarse de forma accesible y actualizada.

RNF.4 – Rendimiento

- **RNF.4.1** – El sistema debe responder de forma fluida ante las acciones más comunes (crear apuesta, apostar, cargar listado).
- **RNF.4.2** – Las consultas a base de datos deben estar optimizadas mediante paginación o filtros cuando sea necesario.

RNF.5 – Escalabilidad y mantenibilidad

- **RNF.5.1** – El sistema debe estar diseñado de forma modular (separación clara entre backend, frontend y base de datos), permitiendo añadir nuevas funcionalidades (como los juegos de casino) sin reestructurar el código existente.
- **RNF.5.2** – El uso de estándares como REST en las APIs y tecnologías ampliamente adoptadas (Spring Boot, Angular) facilita el mantenimiento y la incorporación de nuevos desarrolladores al proyecto.

RNF.6 – Compatibilidad y despliegue

- **RNF.6.1** – La plataforma debe ser accesible desde los principales navegadores web modernos.
- **RNF.6.2** – El sistema debe estar preparado para ser desplegado en un entorno Linux mediante contenedores Docker.
- **RNF.6.3** – La automatización del despliegue mediante GitHub Actions debe permitir actualizaciones sencillas y controladas.

3 Diseño del sistema

3.1 Diseño preliminar

3.1.1. Diagrama de Casos de Uso

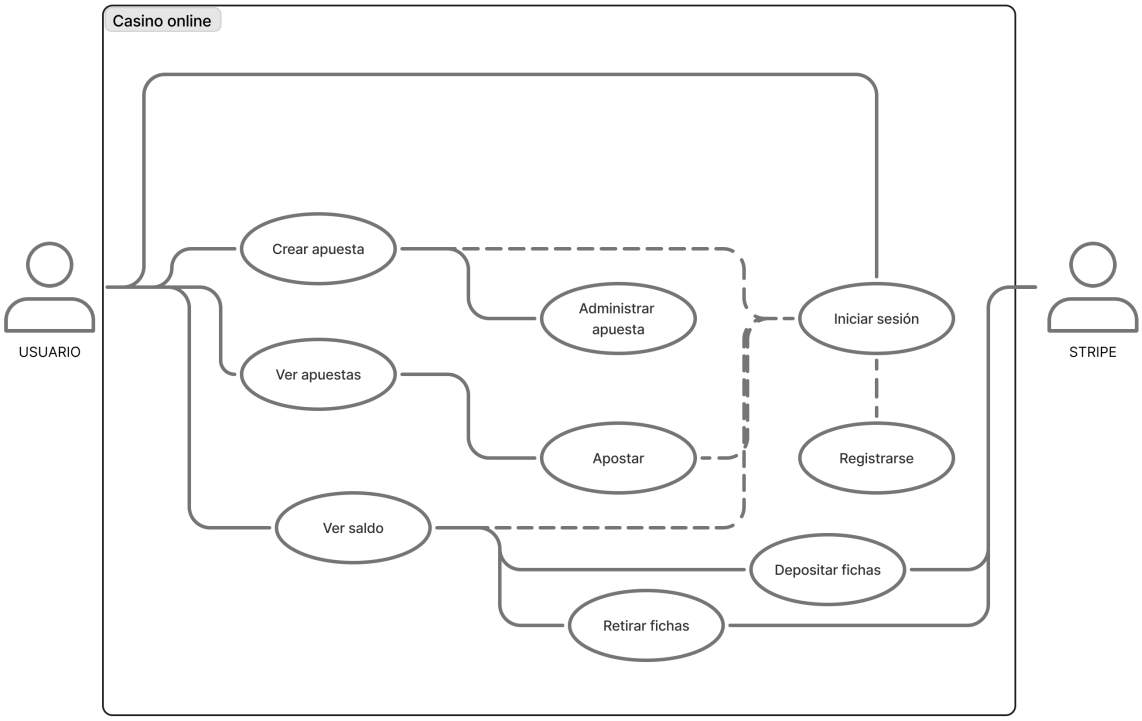

Figura 3: Diagrama de casos de uso del sistema Octobets

El siguiente diagrama de casos de uso representa las principales interacciones entre los actores del sistema y la plataforma **Octobets**. Los actores identificados son:

- **Usuario:** Puede ser un visitante anónimo o un usuario registrado. Tiene acceso a diferentes funcionalidades según su estado de autenticación.
- **Stripe:** Servicio externo utilizado para la gestión de los pagos en la plataforma, concretamente para el depósito y retiro de fichas.

Las funcionalidades principales del sistema se agrupan en los siguientes casos de uso:

- **Ver apuestas:** Cualquier visitante, registrado o no, puede consultar el listado público de apuestas activas.
- **Registrarse / Iniciar sesión:** Permiten a un usuario crear una cuenta o acceder a una existente, lo que desbloquea funcionalidades adicionales.
- **Crear apuesta:** Un usuario autenticado puede generar una nueva apuesta, definiendo sus parámetros básicos.
- **Apostar en apuesta:** Los usuarios registrados pueden participar en apuestas activas, eligiendo la opción que consideren ganadora.
- **Administrar apuesta:** Los creadores de apuestas y los administradores tienen la capacidad de modificar o eliminar sus apuestas.
- **Ver saldo:** Los usuarios autenticados pueden consultar el número de fichas disponibles en su cuenta.
- **Depositar fichas:** Mediante la integración con Stripe, los usuarios pueden realizar ingresos para convertir dinero real en fichas virtuales dentro de la plataforma.
- **Retirar fichas:** Permite al usuario solicitar la retirada de sus fichas disponibles.

El diagrama también refleja, mediante líneas discontinuas, qué acciones requieren estar autenticado. Por ejemplo, acciones como apostar o administrar apuestas solo están disponibles una vez iniciada la sesión.

3.1.2. Diagrama de secuencia

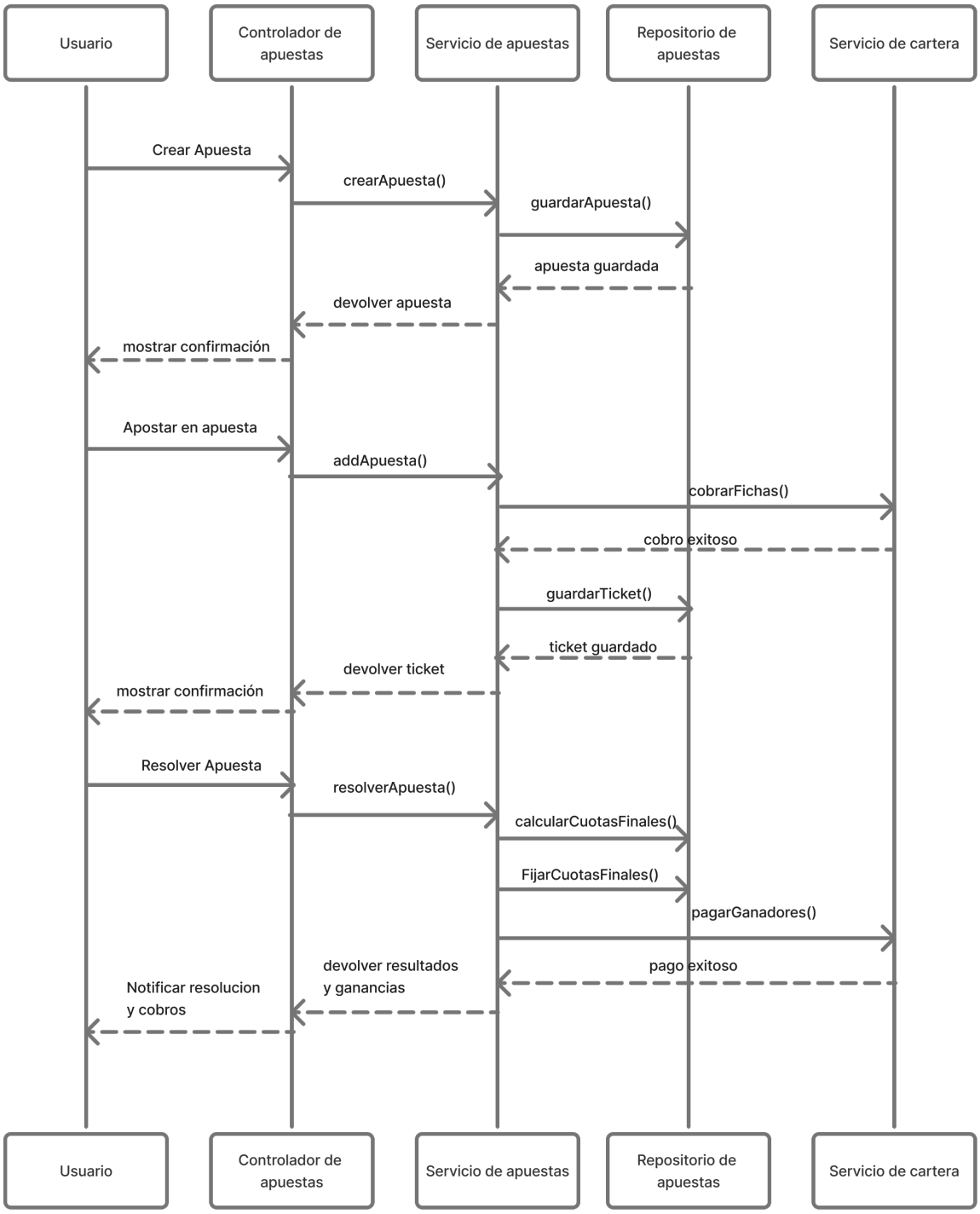

Figura 4: Diagrama de secuencia de los principales flujos del ciclo de vida de una apuesta

El siguiente diagrama de secuencia ilustra el flujo de interacción entre los diferentes componentes del sistema durante el ciclo de vida de una apuesta en la plataforma **Octobets**. Se representan tres casos principales: creación de una apuesta, participación en una apuesta y cierre de la misma.

Participantes (lifelines):

- **Usuario:** Interactúa a través de la interfaz para crear, participar y cerrar apuestas.
- **Controlador de apuestas:** Gestiona las peticiones HTTP y las redirige al servicio correspondiente.
- **Servicio de apuestas:** Contiene la lógica de negocio relacionada con las apuestas.
- **Repositorio de apuestas:** Interactúa con la base de datos para guardar, recuperar y actualizar información.
- **Servicio de cartera:** Gestiona las transacciones de fichas, tanto para cobros como para pagos.

1. Crear apuesta: El usuario inicia el proceso enviando una solicitud de creación. El controlador la redirige al servicio, que a su vez persiste la apuesta mediante el repositorio. Una vez guardada, se devuelve una confirmación al usuario.

2. Apostar en una apuesta existente: El usuario realiza una apuesta. El servicio de apuestas primero solicita al servicio de cartera que cobre las fichas necesarias. Si el cobro es exitoso, se guarda el ticket de participación y se devuelve una confirmación.

3. Resolver apuesta: Cuando la apuesta finaliza, el creador solicita su resolución. El servicio calcula las cuotas finales, las fija, y procede al pago a los ganadores a través del servicio de cartera. Finalmente, se notifican los resultados y posibles ganancias al usuario.

Este diagrama refleja claramente cómo se distribuyen las responsabilidades entre los distintos componentes de la arquitectura, garantizando la separación de lógica de negocio, persistencia y servicios financieros.

3.1.3. Diagrama de clases

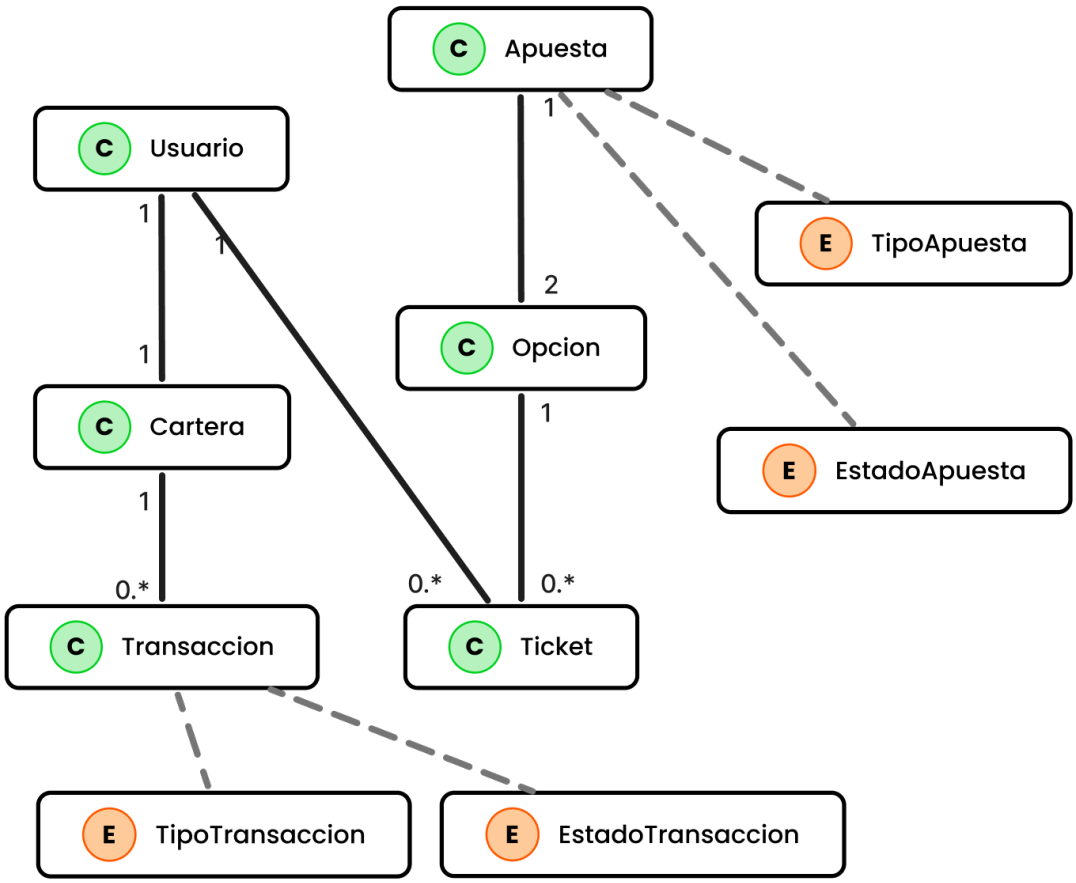

Figura 5: Diagrama de clases del backend de Octobets

El diseño de clases de la aplicación **backend** de **Octobets** sigue una arquitectura por capas bien organizada, separando claramente la lógica de negocio, acceso a datos, controladores REST y seguridad. A continuación se describen los paquetes principales y los tipos de clases que los componen:

model.entity Contiene las clases que representan entidades persistentes del sistema. Cada clase está anotada con `@Entity` y mapeada a una tabla de base de datos. Las relaciones entre entidades están definidas mediante anotaciones como `@OneToOne`, `@OneToMany` y `@ManyToOne`.

Entidades incluidas:

- Usuario
- Cartera
- Apuesta
- Opcion
- Ticket
- Transaccion

Relaciones destacadas:

- Un **Usuario** tiene una **Cartera**.
- Un **Usuario** puede tener múltiples **Ticket**.
- Una **Cartera** puede registrar muchas **Transaccion**.
- Una **Apuesta** puede tener múltiples **Opcion**.
- Cada **Opcion** puede tener múltiples **Ticket**.

Enumeraciones (*enum*) Además de las clases de entidad, el sistema define varios tipos enumerados para representar estados y tipos predefinidos:

- **EstadoApuesta:** posibles estados de una apuesta (por ejemplo, `ABIERTA`, `CERRADA`, `RESUELTA`).
- **TipoApuesta:** identifica si la apuesta es pública, privada, entre amigos, etc.

- **EstadoTransaccion:** refleja el estado de una operación económica (ej. `PENDIENTE`, `COMPLETADA`, `CANCELADA`).
- **TipoTransaccion:** diferencia entre tipos como ingreso o retirada.

Estas enumeraciones están directamente referenciadas en las entidades `Apuesta` y `Transaccion`, y aparecen en el diagrama conectadas con líneas discontinuas para representar la relación de tipo.

repository.dao Contiene las interfaces que extienden de `JpaRepository`, lo que permite realizar operaciones CRUD de forma automática sobre las entidades. Ejemplos:

- UsuarioRepository
- ApuestaRepository
- TicketRepository

service Incluye la lógica de negocio agrupada en interfaces (`XxxService`) e implementaciones (`XxxServiceImpl`). Cada servicio se encarga de gestionar una parte del dominio de la aplicación. Ejemplos:

- ApuestaServiceImpl: lógica para crear, cerrar y listar apuestas.
- UsuarioServiceImpl, CarteraServiceImpl, etc.

También incluye servicios auxiliares como `AuthServiceImpl` y `UsuarioDetailsServiceImpl` para autenticación y carga de usuarios desde base de datos.

rest.controller Aquí se definen los controladores REST, cada uno asociado a una funcionalidad principal. Utilizan `@RestController` para exponer endpoints HTTP. Ejemplos:

- ApuestaRestController, TicketRestController, CarteraRestController.

security Implementa la lógica de autenticación basada en tokens JWT:

- JwtFilter: intercepta y valida peticiones entrantes.
- JwtProvider: genera y valida JWTs.
- UsuarioDetails: implementa `UserDetails` para integrarse con Spring Security.

Este diseño favorece la escalabilidad y la mantenibilidad del sistema, separando claramente responsabilidades y utilizando patrones estándar de desarrollo en Spring Boot.

3.2 Diseño detallado

3.2.1. Diagrama de componentes

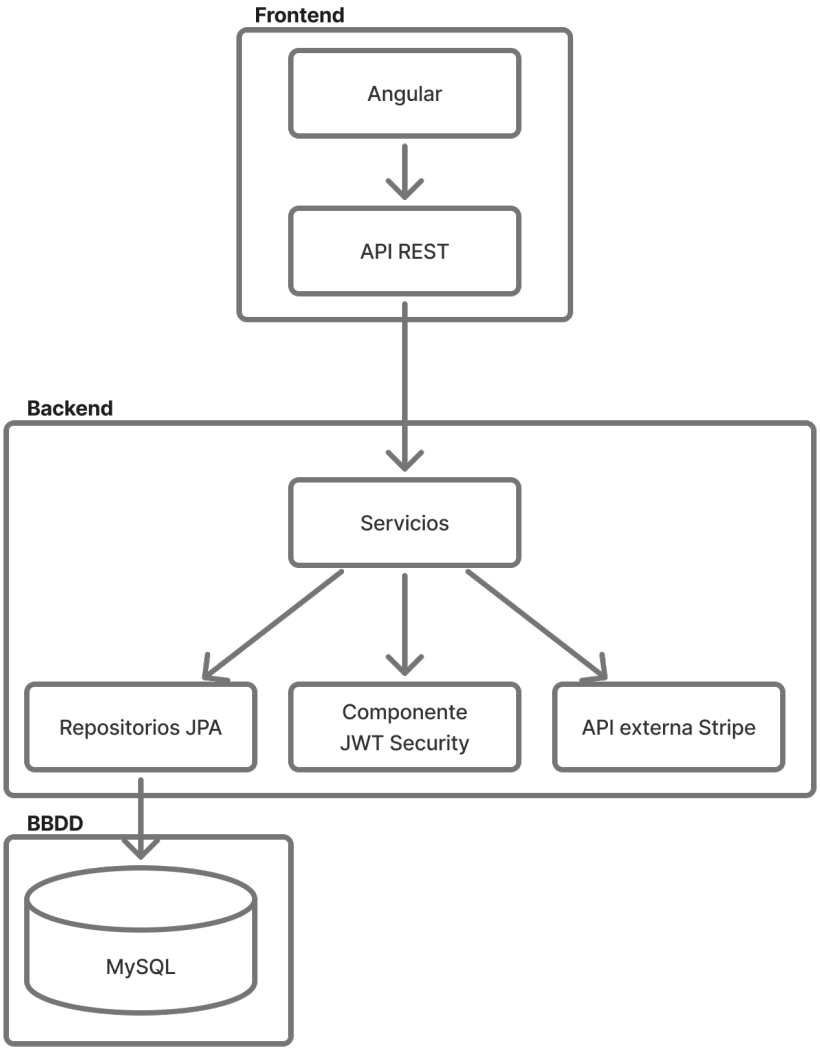

Figura 6: Diagrama de componentes del sistema Octobets

Este diagrama muestra los principales módulos funcionales que componen el sistema **Octobets** y cómo interactúan entre sí.

El **frontend**, desarrollado en Angular, interactúa con el **backend** a través de una API REST. Esta API es el punto de entrada a los servicios del sistema, que se encargan de gestionar la lógica de negocio.

Dentro del backend, los servicios se apoyan en:

- **Repositorios JPA**, que permiten acceder a la base de datos de forma abstracta.
- **Componente JWT Security**, encargado de autenticar y autorizar a los usuarios.
- **API externa de Stripe**, usada para procesar pagos reales y convertirlos en fichas virtuales.

Los datos se almacenan en una base de datos **MySQL**, gestionada a través de los repositorios.

Esta arquitectura favorece la separación de responsabilidades y permite un mantenimiento más sencillo y modular.

3.2.2. Diagrama de despliegue

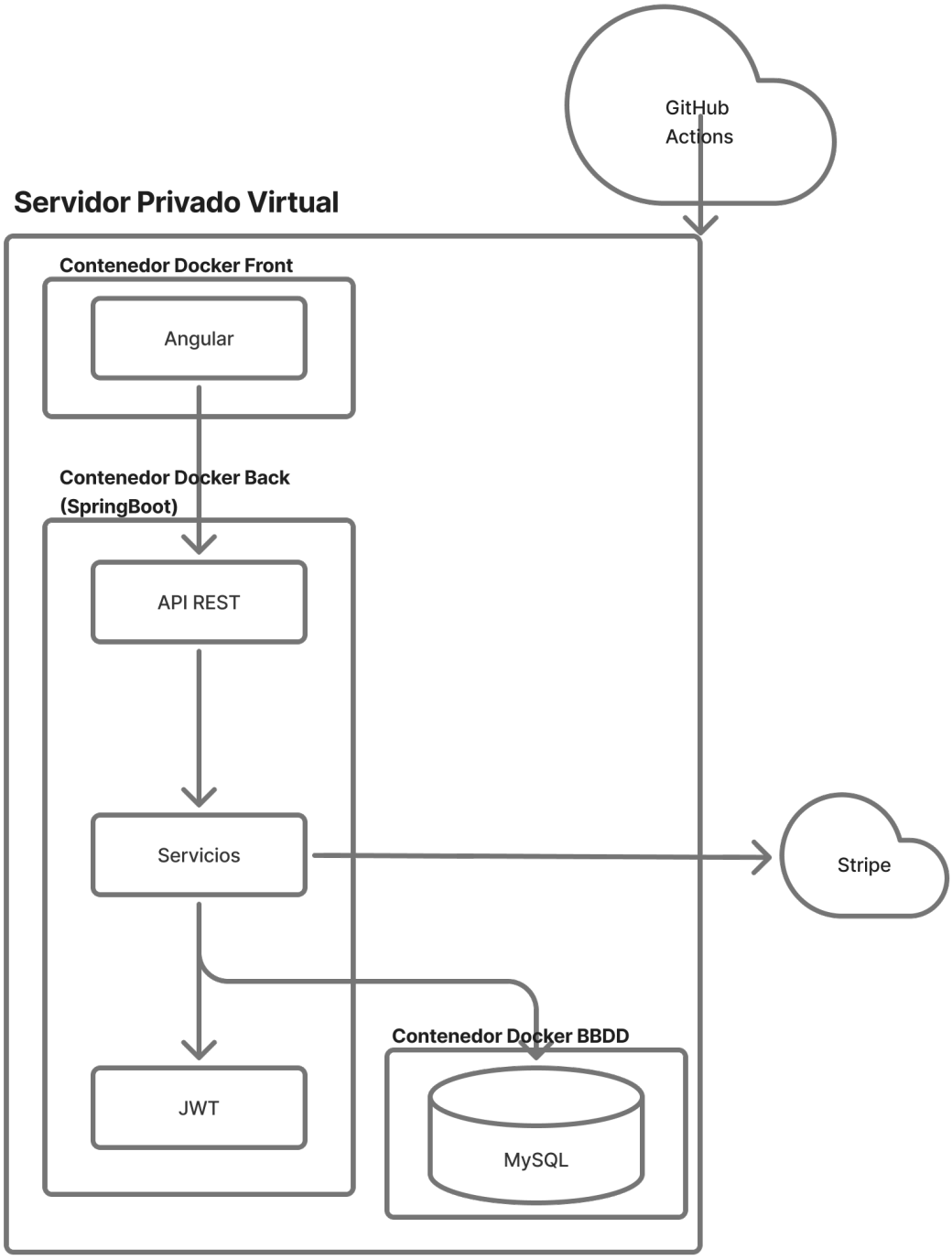

Figura 7: Diagrama de despliegue del sistema Octobets

El siguiente diagrama representa cómo se distribuyen físicamente los distintos componentes del sistema **Octobets** en el entorno de producción.

Tanto el **frontend** (Angular) como el **backend** (Spring Boot) están desplegados en un **VPS** con dominio configurado, lo que permite el acceso público a través de internet. Para ello, se ha utilizado **Docker** para contenerizar los servicios y facilitar su gestión, aislamiento y portabilidad.

El proceso de construcción y despliegue se automatiza mediante **GitHub Actions**, lo que permite una integración y entrega continua (*CI/CD*). Cada vez que se sube una nueva versión al repositorio, se genera automáticamente una nueva imagen Docker y se actualiza el entorno de producción en el VPS.

La base de datos **MySQL** también se encuentra desplegada en el mismo servidor, ejecutándose como un contenedor adicional. Esto reduce la latencia en el acceso a los datos y simplifica la configuración del entorno.

Además, el **backend** mantiene una conexión segura con la API externa de **Stripe**, utilizada para gestionar los pagos reales de fichas.

3.2.3. Diagrama de arquitectura

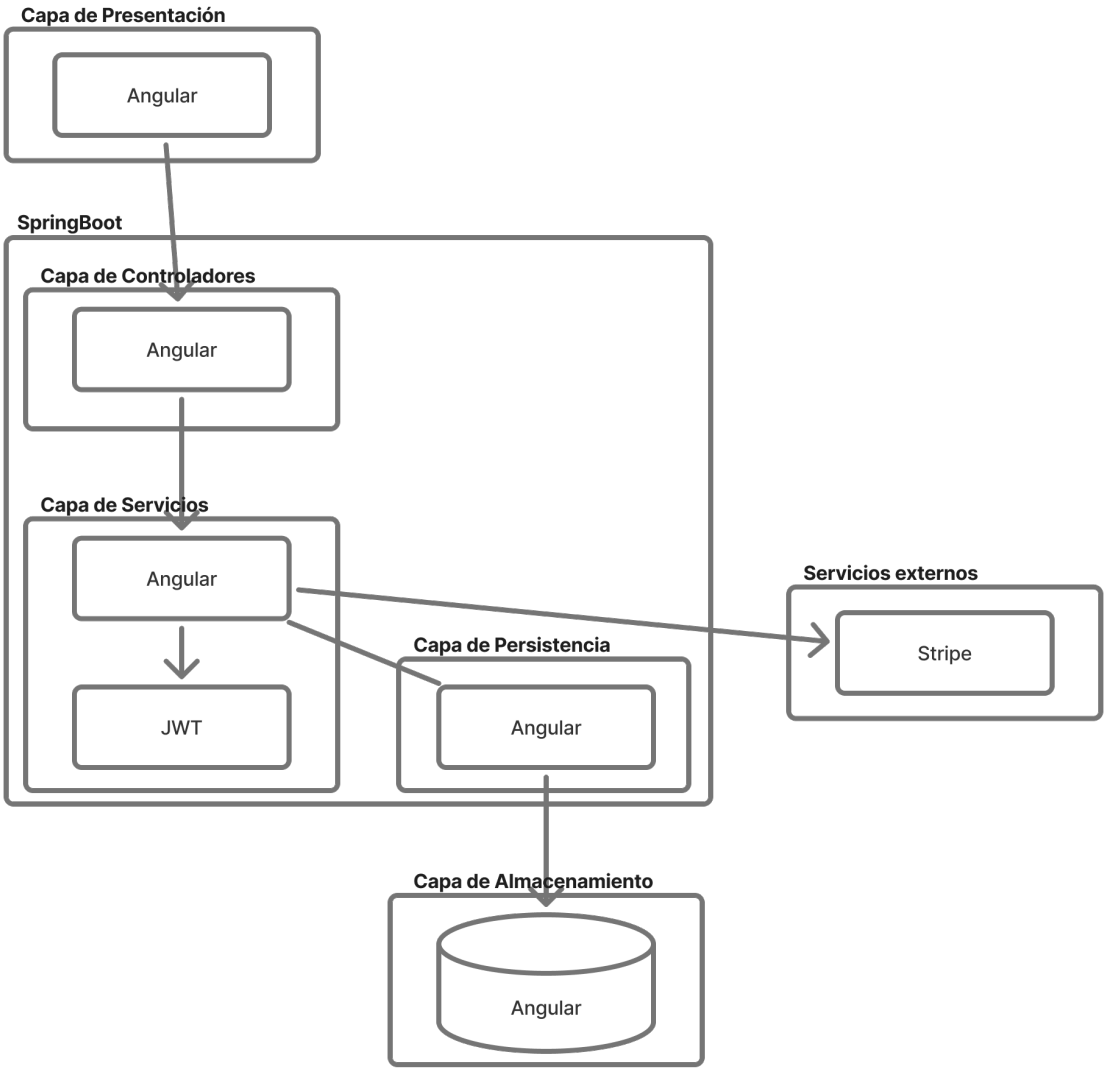

Figura 8: Arquitectura lógica por capas del sistema Octobets

El siguiente diagrama representa la arquitectura lógica del sistema **Octobets**, organizada en capas claramente diferenciadas, siguiendo una estructura tradicional de aplicación web en arquitectura en capas.

- **Capa de Presentación:** desarrollada con Angular, es la interfaz visible para el usuario final. Desde aquí se envían peticiones HTTP al backend a través de una API REST.
- **Capa de Controladores (Spring Boot):** recibe las peticiones del frontend y actúa como puente entre la presentación y la lógica de negocio. Está compuesta por controladores REST anotados con `@RestController`.
- **Capa de Servicios:** aquí se implementa la lógica de negocio de la aplicación. Se encarga de procesar las reglas internas, aplicar validaciones y coordinar las interacciones entre las capas inferior y superior. También incluye el componente de seguridad basado en JWT.
- **Capa de Persistencia:** contiene los repositorios JPA que permiten interactuar con la base de datos de forma abstracta, sin necesidad de escribir SQL manualmente.
- **Capa de Almacenamiento:** representa la base de datos relacional MySQL, donde se guardan de forma persistente todos los datos de usuarios, apuestas, fichas y transacciones.
- **Servicios externos:** la plataforma se conecta con Stripe para la gestión de pagos, lo que permite a los usuarios adquirir fichas a través de transacciones reales.

Esta arquitectura modular favorece la separación de responsabilidades, mejora la mantenibilidad del sistema y permite una evolución escalonada del proyecto.

3.3 Diseño de la Base de Datos

3.3.1. Diagrama Entidad-Relación

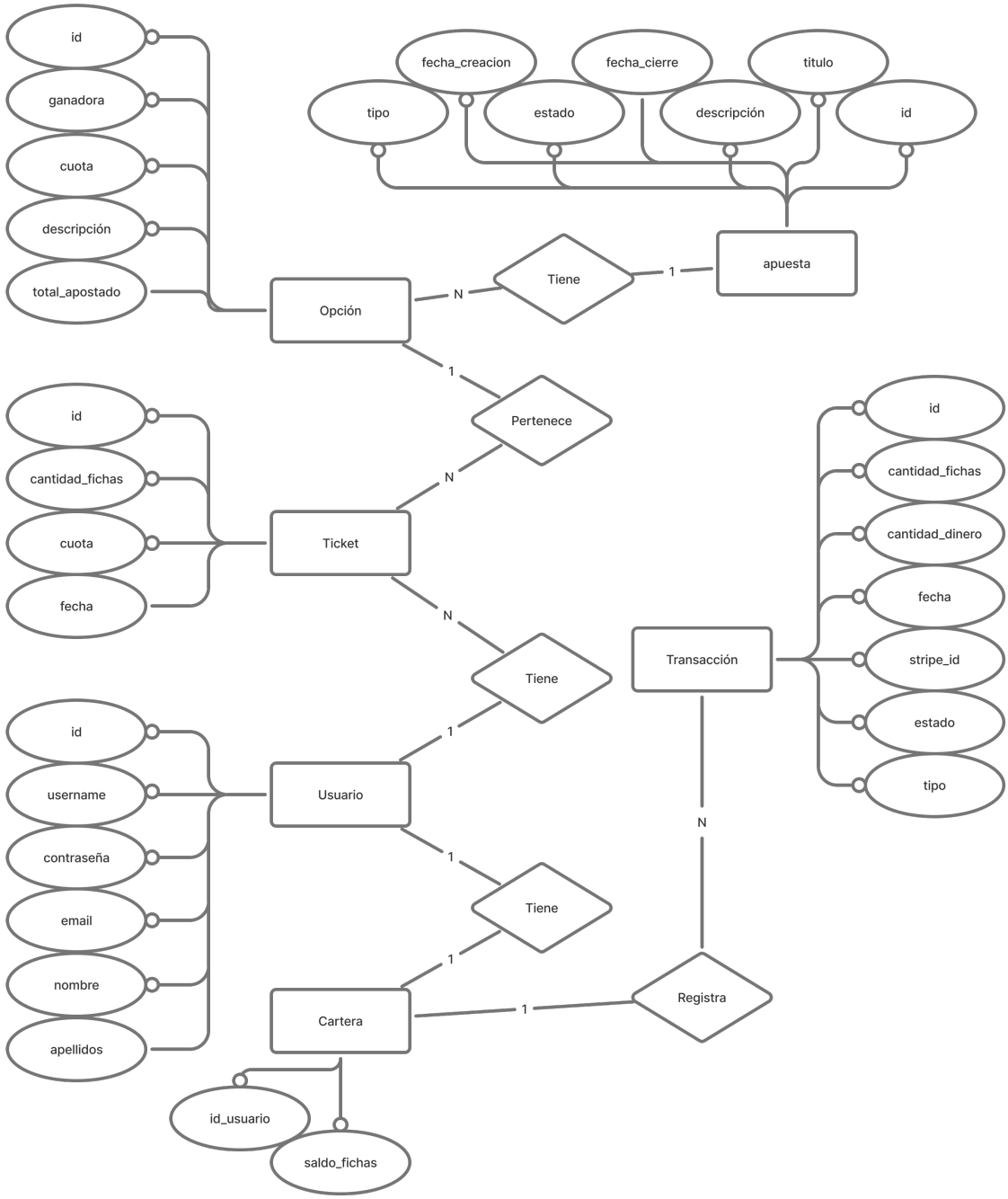

Figura 9: Diagrama Entidad-Relación del sistema Octobets

El siguiente diagrama Entidad-Relación representa la estructura lógica de la base de datos del sistema **Octobets**, detallando las entidades principales, sus atributos y las relaciones entre ellas. El modelo refleja cómo se almacenan y gestionan los datos relacionados con usuarios, apuestas, transacciones y fichas.

Entidades principales:

- **Usuario:** Contiene los datos identificativos y de acceso del usuario (`username`, `contraseña`, `email`, `nombre` y `apellidos`). Cada usuario posee una cartera y puede estar asociado a uno o más tickets.
- **Cartera:** Vinculada de forma uno a uno con un usuario, almacena el número de fichas disponibles (`saldo_fichas`) y registra las transacciones realizadas.
- **Transacción:** Representa cada operación económica del sistema, incluyendo depósito y retirada de fichas. Contiene información como la cantidad de fichas, cantidad de dinero, fecha, identificador de Stripe, estado y tipo (ingreso o retirada). Cada cartera registra múltiples transacciones (relación 1:N).
- **Apuesta:** Define una apuesta creada por un usuario. Sus atributos incluyen título, descripción, fecha de creación y cierre, tipo y estado. Cada apuesta tiene múltiples opciones (relación 1:N).
- **Opción:** Representa cada posible resultado dentro de una apuesta. Incluye atributos como descripción, cuota, si ha resultado ganadora y el total apostado. Cada opción está relacionada con una apuesta y puede estar asociada a varios tickets.
- **Ticket:** Recoge la participación de un usuario en una apuesta concreta a través de una opción específica. Contiene los atributos `cantidad_fichas`, `cuota` y `fecha`. Un ticket pertenece a una opción y está vinculado a un usuario.

3.3.2. Modelo relacional (Nivel lógico)

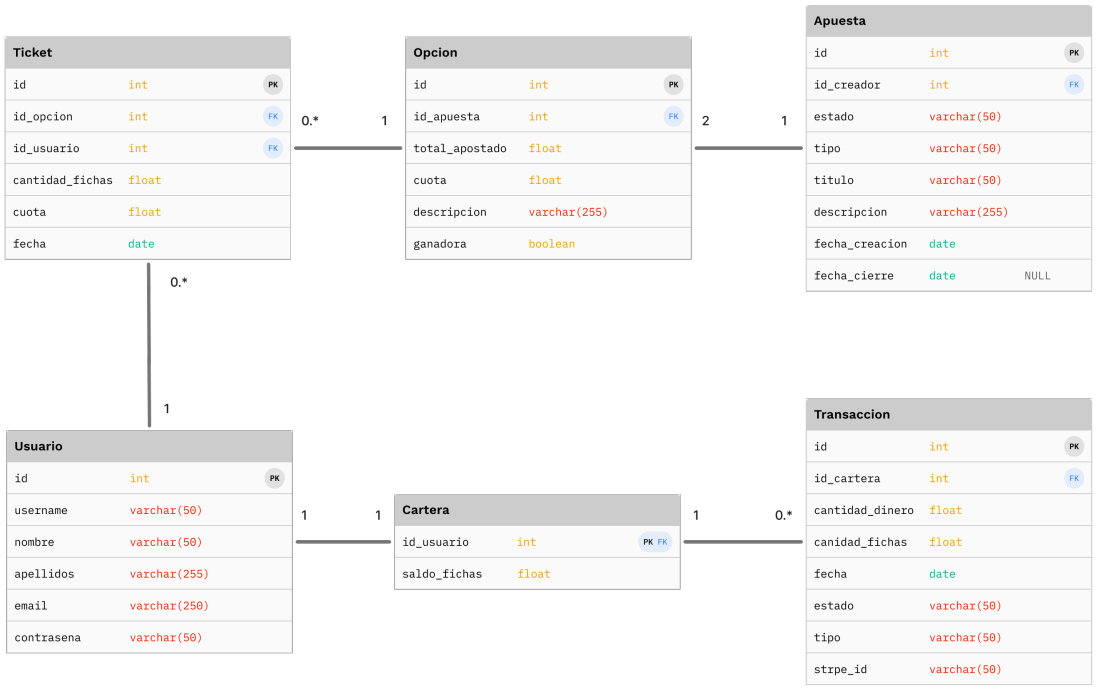

Figura 10: Modelo relacional de la base de datos Octobets (nivel lógico)

El siguiente modelo relacional representa la estructura lógica de la base de datos utilizada en **Octobets**, mostrando las entidades principales, sus atributos y relaciones.

Entidades principales:

- **Usuario:** Contiene los datos personales y de autenticación de los usuarios registrados. Relación 1:1 con la entidad **Cartera** y 1:N con **Ticket**.
- **Cartera:** Representa el saldo de fichas asociado a cada usuario. Cada cartera pertenece a un único usuario y puede tener múltiples transacciones asociadas.

- **Apuesta:** Define las apuestas creadas por los usuarios. Incluye atributos como título, tipo, estado y fechas. Relación 1:N con la entidad **Opcion**.
- **Opcion:** Representa una posible respuesta dentro de una apuesta. Cada opción pertenece a una apuesta y puede estar vinculada a múltiples **Ticket**.
- **Ticket:** Almacena la información de la participación de un usuario en una opción concreta, incluyendo la cantidad apostada y la cuota.
- **Transaccion:** Registra las operaciones económicas de los usuarios (ingresos y retiradas de fichas), asociadas a una cartera concreta. Contiene también el `stripe_id` para el seguimiento de pagos.

Este modelo relacional, junto con el diagrama ER, permite gestionar de forma eficiente la información necesaria para operar la plataforma, asegurando la integridad de los datos mediante claves primarias y foráneas, y estructurando las relaciones de manera clara y normalizada.

3.3.3. Nivel físico (Tablas)

El nivel físico describe la estructura definitiva de cada tabla MySQL que conforma la base de datos de **Octobets**. Se detallan campos, tipos y restricciones de clave primaria (PK) o foránea (FK).

Cuadro 1: **Tabla** `usuario`

Campo	Descripción	Tipo MySQL	Restricciones
id	Identificador único de usuario	INT	PK, AUTO_INCREMENT, NOT NULL
username	Nombre de usuario	VARCHAR(50)	UNIQUE, NOT NULL
nombre	Nombre real del usuario	VARCHAR(50)	NOT NULL
apellidos	Apellidos del usuario	VARCHAR(255)	NOT NULL
email	Correo electrónico	VARCHAR(250)	UNIQUE, NOT NULL
contrasena	Contraseña cifrada	VARCHAR(50)	NOT NULL

Cuadro 2: **Tabla** `cartera`

Campo	Descripción	Tipo MySQL	Restricciones
id_usuario	ID del usuario propietario	INT	PK, FK → usuario(id), NOT NULL
saldo_fichas	Cantidad de fichas disponibles	FLOAT	DEFAULT 0, NOT NULL

Cuadro 3: **Tabla** `apuesta`

Campo	Descripción	Tipo MySQL	Restricciones
id	Identificador de apuesta	INT	PK, AUTO_INCREMENT, NOT NULL
id_creador	Usuario que la creó	INT	FK → usuario(id), NOT NULL
estado	Estado actual (abierta, cerrada...)	VARCHAR(50)	NOT NULL
tipo	Tipo de apuesta	VARCHAR(50)	NOT NULL
titulo	Título de la apuesta	VARCHAR(50)	NOT NULL
descripcion	Descripción de la apuesta	VARCHAR(255)	NOT NULL
fecha_creacion	Fecha de creación	DATE	NOT NULL
fecha_cierre	Fecha límite de participación	DATE	NULL

Cuadro 4: **Tabla** `opcion`

Campo	Descripción	Tipo MySQL	Restricciones
id	Identificador de opción	INT	PK, AUTO_INCREMENT, NOT NULL
id_apuesta	Apuesta a la que pertenece	INT	FK → apuesta(id), NOT NULL
total_apostado	Total de fichas apostadas	FLOAT	DEFAULT 0, NOT NULL
cuota	Cuota definida por el creador	FLOAT	NOT NULL
descripcion	Descripción de la opción	VARCHAR(255)	NOT NULL
ganadora	Indica si fue la opción ganadora	BOOLEAN	DEFAULT FALSE, NOT NULL

Cuadro 5: **Tabla** `ticket`

Campo	Descripción	Tipo MySQL	Restricciones
id	Identificador del ticket	INT	PK, AUTO_INCREMENT, NOT NULL
id_opcion	Opción seleccionada	INT	FK → opcion(id), NOT NULL
id_usuario	Usuario que realizó la apuesta	INT	FK → usuario(id), NOT NULL
cantidad_fichas	Fichas apostadas	FLOAT	NOT NULL
cuota	Cuota fijada al hacer la apuesta	FLOAT	NOT NULL
fecha	Fecha en la que se realizó	DATE	NOT NULL

Cuadro 6: **Tabla** `transaccion`

Campo	**Descripción**	**Tipo MySQL**	**Restricciones**
id	Identificador de transacción	INT	PK, AUTO_INCREMENT, NOT NULL
id_cartera	Cartera asociada	INT	FK → cartera(id_usuario), NOT NULL
cantidad_dinero	Importe en euros	FLOAT	NOT NULL
cantidad_fichas	Número de fichas obtenidas	FLOAT	NOT NULL
fecha	Fecha de la operación	DATE	NOT NULL
estado	Estado del pago (ej: COMPLETADO)	VARCHAR(50)	NOT NULL
tipo	Tipo de operación (ingreso/retiro)	VARCHAR(50)	NOT NULL
stripe_id	ID devuelto por Stripe	VARCHAR(255)	NOT NULL

3.4 Guía de estilos e iconografía

3.4.1. Guía de estilos

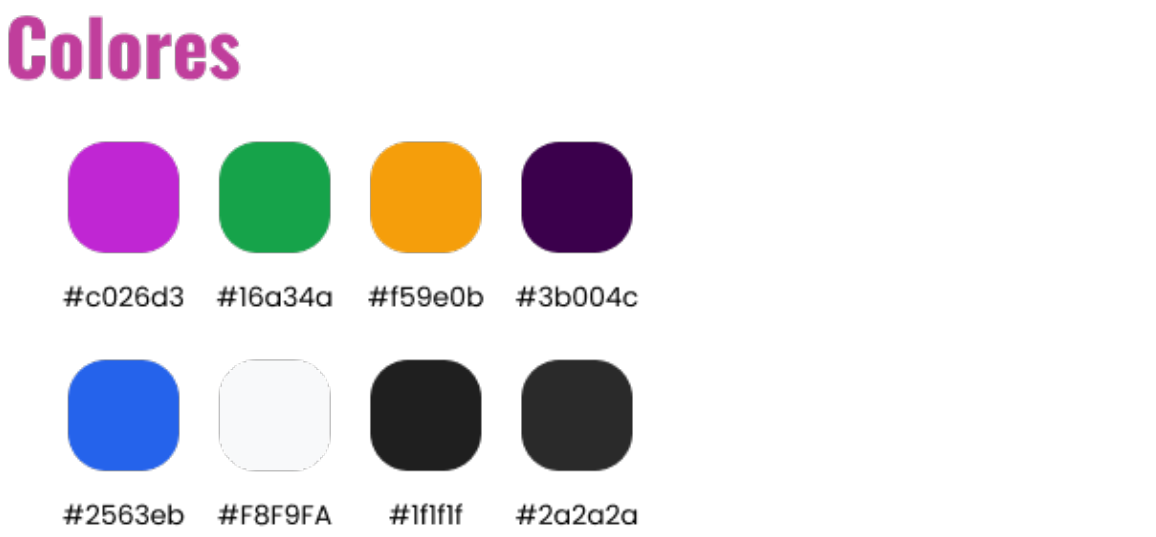

Figura 11: Guía visual aplicada al diseño de Octobets

El diseño de **Octobets** combina una estética moderna con una interfaz limpia y funcional. Se han utilizado tecnologías como **Tailwind CSS** para definir una interfaz modular y coherente, y un enfoque visual tipo *bento*, en el que los bloques de contenido están organizados como tarjetas o secciones independientes.

Esta estructura mejora la legibilidad, la jerarquía visual y la experiencia de usuario.

Además, se ha mantenido un espaciado constante de **30px** entre bloques, garantizando separación, limpieza visual y coherencia a lo largo de toda la aplicación.

3.4.2. Iconografía e ilustraciones

El sistema combina iconografía funcional y recursos gráficos personalizados para ofrecer una experiencia visual más rica y coherente.

Iconos del sistema Se han utilizado **Material Icons** de Google para representar acciones básicas dentro de la interfaz. Gracias a su integración nativa en Angular mediante el componente `<mat-icon>`, su uso ha sido sencillo y versátil. Ejemplos comunes incluyen:

- `check_circle`, `close`, `help_outline`, `expand_more`, entre otros.

Estos iconos se aplican en botones, formularios y menús, manteniendo una línea visual coherente con el resto de la interfaz.

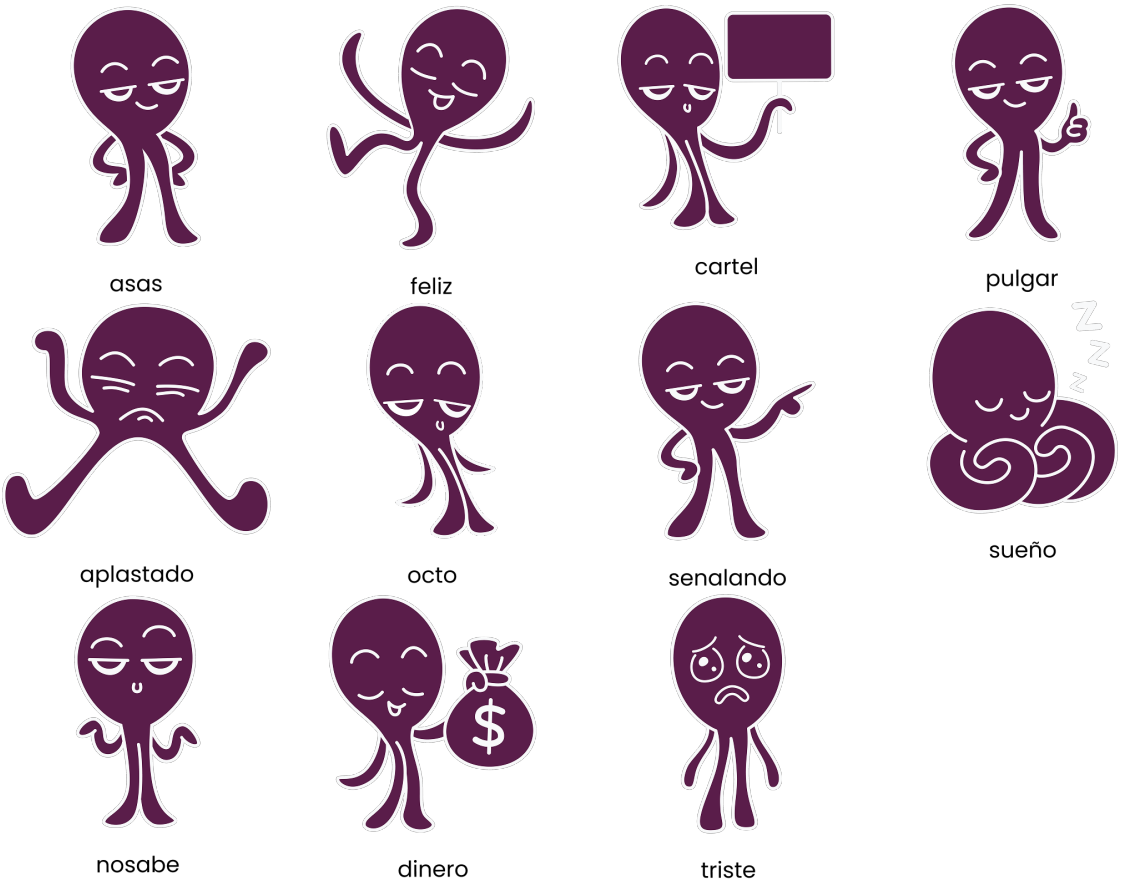

Figura 12: Ilustraciones de Octobets

Ilustraciones personalizadas Además de los iconos funcionales, se han creado ilustraciones personalizadas basadas en la mascota de la marca: un simpático personaje con forma de pulpo. Estas imágenes se han utilizado para apoyar visualmente mensajes clave y reforzar la identidad de **Octobets**.

Cuadro 7: Ilustraciones implementadas en la aplicación

Ilustración	Uso en la aplicación
`aplastado.png`	Página de not-found (404)
`asas.png`	Página de login
`senalando.png`	Pantalla de registro
`dinero.png`	Página de gestión económica (fichas/dinero)
`feliz.png`	Página principal/home
`nosabe.png`	Sección ¿Cómo funciona?
`octo.png`	Logotipo principal en cabecera y login
`triste.png`	Mensajes de error o contenido no disponible

Otras ilustraciones como `pulgar.png`, `sueno.png` o `cartel.png` han sido diseñadas, pero no implementadas aún en la aplicación. No obstante, su estilo y temática las hacen perfectamente reutilizables para futuras ampliaciones del proyecto.

Este conjunto gráfico propio ayuda a diferenciar la plataforma de otras similares, aportando personalidad y cercanía al diseño de la interfaz.

4 Codificación de la solución

4.1 Planteamiento de la estrategia de programación

Para el desarrollo de **Octobets** se optó por una metodología de trabajo ágil, tomando como base el marco **Scrum**, por su capacidad para estructurar el proyecto en fases cortas y permitir iteraciones rápidas. Esta elección facilitó una evolución progresiva del producto, especialmente útil al haberse producido un reajuste del alcance durante el proceso.

El desarrollo se organizó en bloques de trabajo que permitieron alternar entre la implementación del *backend* (Spring Boot) y el *frontend* (Angular). Se fueron cerrando funcionalidades completas de forma progresiva: gestión de usuarios, creación y participación en apuestas, integración con Stripe, etc. El trabajo se estructuró de forma que cada módulo funcional se desarrollaba, probaba e integraba por separado, permitiendo validar resultados de forma

continua.

Para el control del código fuente se utilizó **Git** a través de **GitHub**, donde se mantuvo un repositorio privado que incluía ramas diferenciadas para organizar tareas por componente. Posteriormente se configuró un flujo automatizado de despliegue mediante **GitHub Actions**, permitiendo actualizar el servidor tras cada nueva versión estable.

Como entregable final, el repositorio se publica con todo el código fuente dividido en carpetas de `frontend` y `backend`, así como el `init.sql` necesario para generar la estructura de base de datos.

4.2 Especificaciones técnicas para el desarrollo

4.2.1. Requerimientos técnicos de hardware

Durante el desarrollo de **Octobets** se utilizaron dos equipos personales: un portátil y un ordenador de sobremesa, ambos con sistema operativo **Windows 11** y especificaciones suficientes para llevar a cabo tareas de programación, compilación y pruebas de manera fluida. Ambos contaban con al menos **8 GB de RAM** y **almacenamiento SSD**, lo cual permitió una experiencia de desarrollo ágil y sin cuellos de botella relevantes.

Para el entorno de producción, se optó por el despliegue en un **VPS (Servidor Privado Virtual)** con dominio propio configurado. El servidor cuenta con los siguientes recursos aproximados:

- 1 vCPU
- 4 GB de RAM
- 50 GB de almacenamiento SSD

Este entorno ha sido suficiente para alojar tanto el *backend* (Spring Boot) como el *frontend* (Angular), así como la base de datos MySQL y los contenedores Docker necesarios. Si en el futuro el número de usuarios creciera significativa-

mente, sería posible escalar verticalmente el VPS o migrar a una solución en la nube más flexible.

4.2.2. Requerimientos técnicos de software

Durante el desarrollo de **Octobets** se ha trabajado con un conjunto de herramientas modernas y tecnologías ampliamente utilizadas en el desarrollo web *full-stack*. A continuación, se detallan los principales recursos de software utilizados:

- **Sistema operativo:** Windows 11 en ambos equipos de desarrollo.
- **Backend:**
  - Java (JDK 21)
  - Spring Boot (estructura modular, API REST)
  - Spring Security y JWT para la autenticación y autorización.
- **Frontend:**
  - Angular 19
  - Angular CLI para generación de componentes y servicios.
  - Tailwind CSS como framework de estilos utilitario.
- **Base de datos:**
  - MySQL 8 como sistema gestor de base de datos.
  - MySQL Workbench para administración visual.
- **Pagos:**
  - Stripe (mediante integración REST) para la gestión de transacciones económicas dentro de la plataforma.
- **Entorno de desarrollo:**
  - Visual Studio Code como editor principal.

- **Control de versiones y despliegue:**
  - Git y GitHub como sistema de control de versiones.
  - GitHub Actions para automatizar el despliegue continuo en el VPS.
- **Contenedores y despliegue:**
  - Docker para contenerizar backend, frontend y base de datos en producción.
- **Gestión de dependencias:**
  - Maven para el backend.
  - npm para el frontend Angular.

Este *stack* permite una arquitectura limpia y desacoplada, donde cada parte del sistema se desarrolla y despliega de forma independiente, garantizando flexibilidad y facilidad de mantenimiento.

5 Descripción general de la solución creada

A continuación, se presenta un recorrido visual por las principales pantallas de la plataforma **Octobets**, acompañadas de una breve explicación sobre su funcionalidad. Cada sección refleja fielmente el estado actual de desarrollo de la aplicación.

Página de Inicio (Home)

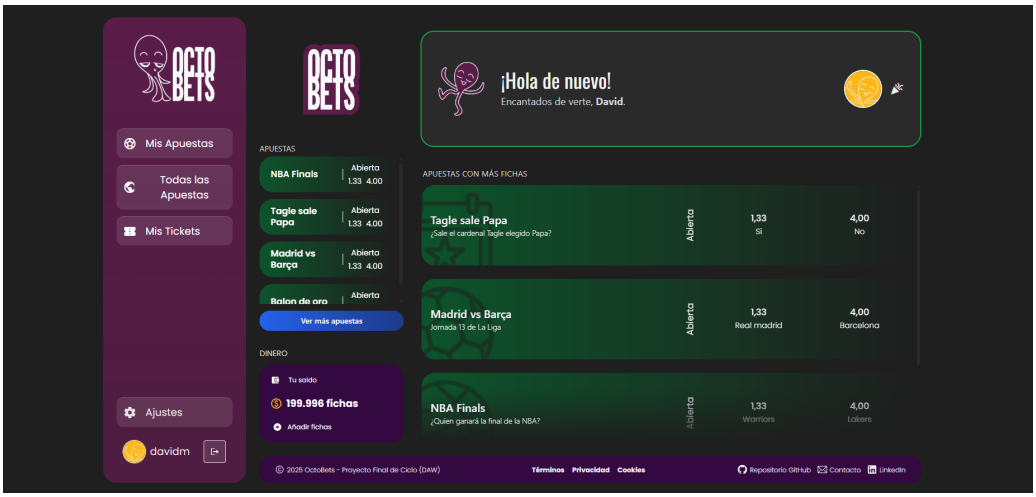

Figura 13: Vista principal de la plataforma para usuarios autenticados y visitantes.

La página de inicio es accesible tanto para usuarios autenticados como no autenticados. Muestra un resumen visual del contenido disponible en la plataforma, especialmente el listado de apuestas activas. Desde aquí se pueden explorar todas las apuestas, aplicar filtros y consultar las cuotas, sin necesidad de iniciar sesión. Para participar en las apuestas o crear nuevas, sí es necesario estar registrado.

Pantalla de Registro

Figura 14: Formulario de registro con validación y diseño visual amigable.

En esta sección, los nuevos usuarios pueden crear una cuenta rellenando sus datos personales. El formulario incluye validación de campos, confirmación de contraseña y feedback visual inmediato ante errores o campos incompletos. La ilustración del pulpo señala el formulario de forma simpática.

Pantalla de Inicio de Sesión

Figura 15: Formulario de login con mensajes claros de error y acceso al registro.

Desde esta vista, los usuarios registrados pueden acceder a la plataforma introduciendo su usuario o correo y contraseña. Se ofrecen mensajes claros ante errores de autenticación y un acceso rápido al registro si aún no se dispone de cuenta.

Sección de Apuestas

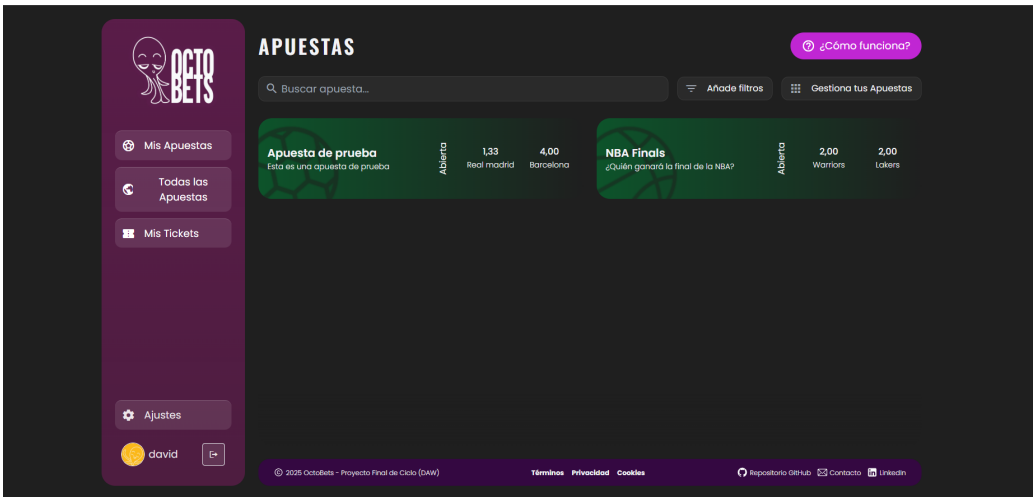

Figura 16: Listado de apuestas públicas con filtros y vista de tarjetas.

En esta pantalla se listan todas las apuestas disponibles, con filtros por estado y tipo. Cada apuesta se representa mediante una tarjeta clara que muestra título, cuotas y opciones. Es accesible tanto para usuarios autenticados como visitantes.

Crear Apuesta

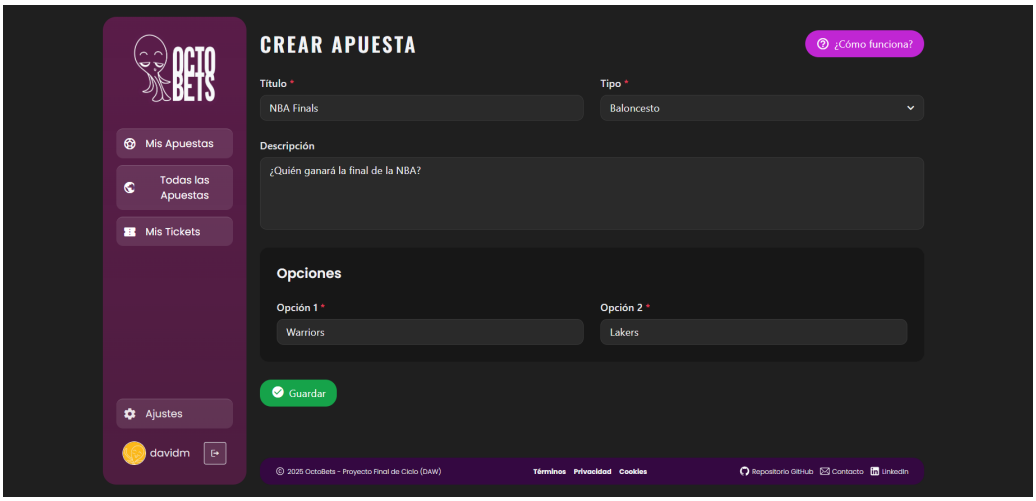

Figura 17: Formulario para crear nuevas apuestas con múltiples opciones.

Los usuarios pueden proponer sus propias apuestas desde esta sección. Se incluye un formulario con campos obligatorios para definir el título, descripción, tipo y opciones. Al guardar, la apuesta queda publicada y visible para otros usuarios.

Gestión de Apuestas (Mis Apuestas)

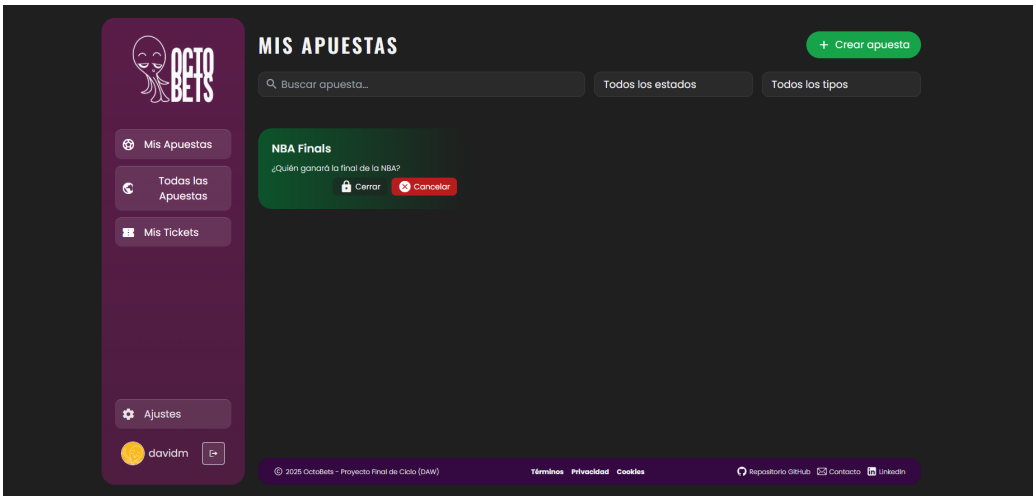

Figura 18: Gestión de apuestas creadas: cerrar, eliminar, consultar.

Aquí se listan las apuestas creadas por el usuario. Desde esta vista puede cerrarlas o cancelarlas. Se presenta de forma visual cada apuesta con opciones de acción rápida, adaptadas a su estado.

Sección de Tickets

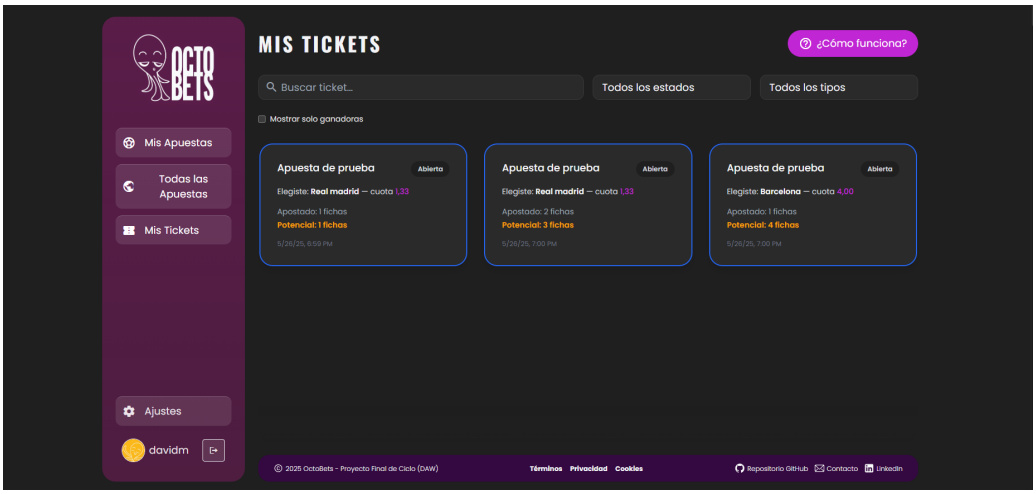

Figura 19: Resumen de apuestas realizadas con detalles y resultados.

Muestra un histórico de apuestas en las que el usuario ha participado. Cada tarjeta de ticket incluye información sobre la apuesta, la opción elegida, la cantidad apostada, la fecha y el beneficio potencial. Los tickets ganadores y perdedores se diferencian visualmente.

Cartera y Transacciones

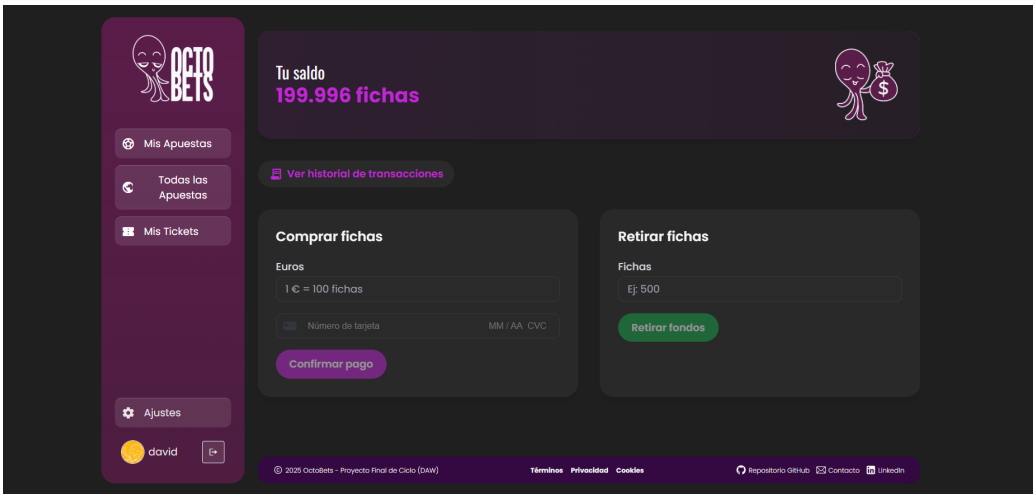

Figura 20: Gestión de saldo del usuario.

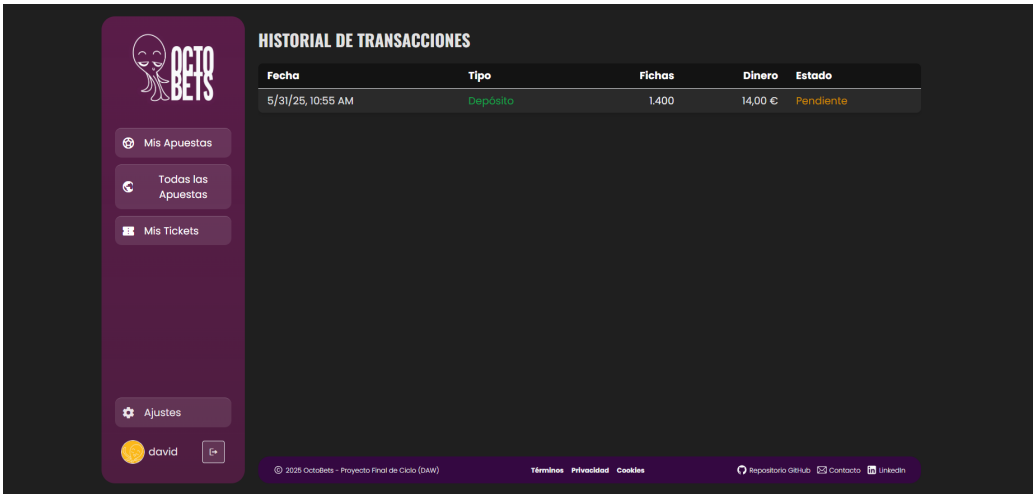

Figura 21: Consulta de movimientos económicos del usuario.

Desde la sección de cartera, el usuario puede ver su saldo actual de fichas y operar para comprar o retirar fondos. Además, puede consultar el historial de transacciones con detalles como fecha, tipo, importe y estado.

Ajustes de Usuario

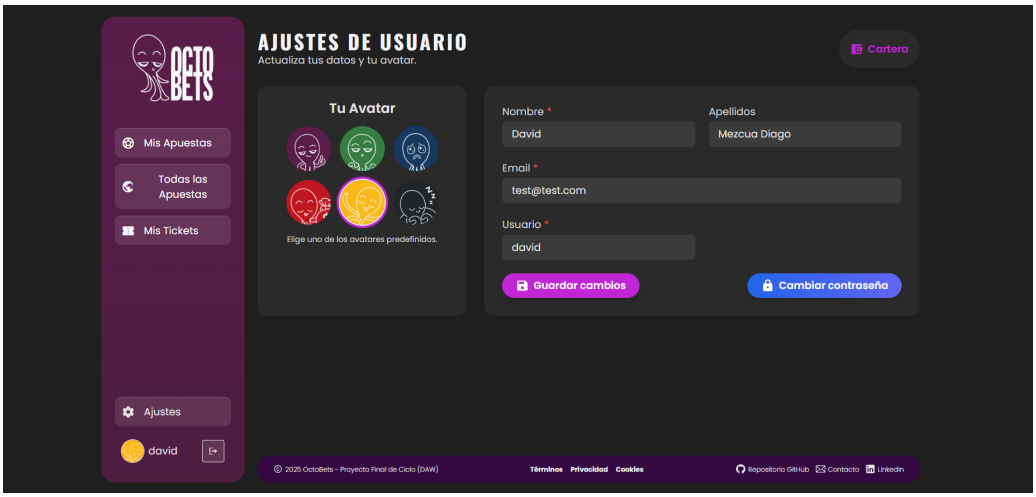

Figura 22: Gestión del perfil personal y cambio de avatar o contraseña.

Permite al usuario actualizar su perfil (nombre, correo, username) y seleccionar un avatar visual representativo entre varias opciones prediseñadas. También puede cambiar su contraseña de forma segura.

Este conjunto de pantallas conforma una experiencia completa de uso, orientada a la simplicidad, accesibilidad y personalización, sin descuidar la seguridad y la claridad en la gestión de apuestas y fondos.

6 Pruebas de Unidad

Durante el desarrollo de la plataforma **OctoBets**, se llevaron a cabo pruebas de unidad en el *backend* con el objetivo de garantizar que los componentes individuales funcionaran correctamente de forma aislada. Estas pruebas se centraron principalmente en los servicios, que constituyen el núcleo de la lógica de negocio de la aplicación.

Las pruebas se implementaron utilizando el framework de testing **JUnit 5**, y

para simular la interacción con los repositorios y otros componentes externos se utilizó **Mockito**, una herramienta ampliamente adoptada para crear objetos simulados (*mocks*) y verificar comportamientos específicos.

Cobertura de pruebas

Las pruebas de unidad desarrolladas abarcan los siguientes aspectos funcionales:

- **Servicios de usuario:** verificación del registro, autenticación y recuperación de datos de usuario.
- **Servicios de apuestas:** creación, cierre y recuperación de apuestas junto a sus opciones asociadas.
- **Servicios de tickets:** validación de las apuestas realizadas por los usuarios y almacenamiento de los tickets generados.
- **Servicios de transacciones:** registro de depósitos y retiros, integración simulada con Stripe y actualización del saldo de fichas en la cartera del usuario.
- **Control de errores:** pruebas con entradas inválidas para comprobar el correcto lanzamiento de excepciones como `ResourceNotFoundException`.

Beneficios obtenidos

Gracias a estas pruebas:

- Se validó que los métodos de los servicios devolvían los resultados esperados en distintos escenarios.
- Se garantizó una correcta interacción entre los servicios y los repositorios JPA, usando simulaciones para evitar dependencias reales de la base de datos.

- Se facilitaron futuras refactorizaciones del código, al disponer de una base de tests que permitía comprobar rápidamente si se introducían errores.
- Se aumentó la robustez general de la aplicación, minimizando el riesgo de fallos en producción.

Estas pruebas de unidad constituyen una parte esencial del aseguramiento de calidad del sistema, y sentaron las bases para un desarrollo más fiable, modular y mantenible.

7 Mantenimiento

Una vez finalizado el desarrollo de la plataforma y realizado el despliegue en un servidor accesible públicamente, se ha contemplado un plan básico de mantenimiento que permita corregir errores, mejorar funcionalidades y adaptar el sistema a posibles necesidades futuras.

Las principales consideraciones de mantenimiento incluyen:

- **Corrección de errores:** se monitoriza el comportamiento de la plataforma a través de logs y reportes manuales, lo que permite detectar y solucionar posibles fallos tras su uso por parte de usuarios reales.
- **Actualización de dependencias:** tanto en el *backend* (Spring Boot, dependencias de seguridad, etc.) como en el *frontend* (Angular, Tailwind, etc.), se prevé realizar actualizaciones periódicas para corregir vulnerabilidades y beneficiarse de mejoras de rendimiento.
- **Mejora de funcionalidades:** se han identificado posibles ampliaciones como la incorporación de estadísticas de usuario, mejoras en la interfaz o nuevas formas de visualizar apuestas, que podrían implementarse en versiones futuras.
- **Escalabilidad y rendimiento:** aunque la aplicación está pensada inicialmente para un entorno pequeño, el uso de contenedores Docker y la arqui-

tectura modular permite adaptarse fácilmente a un entorno más exigente si se detecta un crecimiento en el tráfico.

- **Monitorización y copias de seguridad:** el servidor VPS utilizado permite configurar copias de seguridad automáticas y herramientas de monitorización básicas para garantizar la disponibilidad y seguridad de los datos.
- **Gestión del repositorio:** el código fuente se encuentra versionado en GitHub, lo que facilita la trazabilidad de los cambios y la colaboración futura en caso de ampliar el equipo de desarrollo o retomar el proyecto.

En resumen, el mantenimiento de **OctoBets** ha sido planteado desde una perspectiva práctica, centrada en asegurar la estabilidad de la plataforma y su potencial evolución en función del uso y la retroalimentación que reciba.

8 Futuras Ampliaciones

Aunque el alcance final de **OctoBets** se centró exclusivamente en el sistema de gestión de apuestas personalizadas, durante la fase de diseño inicial se contemplaron funcionalidades más ambiciosas que podrían retomarse y desarrollarse en versiones futuras de la plataforma.

Entre las ampliaciones más relevantes que se han identificado, destacan:

- **Implementación de juegos de casino:** originalmente se planteó incorporar una sección dedicada a juegos como tragamonedas (slots), ruleta o similares, accesibles desde la misma plataforma. La idea era ofrecer una experiencia lúdica más completa. Sin embargo, por limitaciones de tiempo y especialmente por la complejidad legal que implicaría este tipo de funcionalidades, se decidió posponerlo. Este módulo sigue siendo una opción viable a largo plazo, siempre que se estudien y resuelvan las cuestiones normativas correspondientes.
- **Sistema de cierre automático de apuestas:** actualmente, las apuestas deben ser gestionadas manualmente por el administrador. Una posible

mejora sería incorporar una lógica de cierre automático basada en fechas límite o resultados verificados por fuentes externas.

- **Notificaciones en tiempo real:** el uso de WebSockets o tecnologías similares permitiría alertar a los usuarios sobre cambios en las apuestas, nuevos tickets validados o movimientos en su cartera de forma inmediata.
- **Estadísticas de usuario:** un apartado con gráficas y métricas que reflejen el rendimiento de cada usuario (apuestas ganadas, dinero apostado, porcentaje de acierto, etc.) contribuiría a enriquecer la experiencia.
- **Sistema de comentarios o valoraciones:** permitir a los usuarios interactuar a través de comentarios en apuestas públicas podría fomentar el aspecto social de la plataforma.
- **Internacionalización (*i18n*):** la posibilidad de traducir la plataforma a varios idiomas ampliaría su alcance potencial más allá del público hispanohablante.
- **Ampliación de los métodos de pago:** actualmente solo se simula la integración con Stripe. La inclusión de nuevos métodos como PayPal, criptomonedas o monederos virtuales podría aportar mayor flexibilidad al sistema de transacciones.

Estas ampliaciones podrían desarrollarse de forma incremental si el proyecto continúa activo en el futuro o se plantea su publicación como producto comercial o colaborativo.

9 Valoración Económica

Para estimar el coste económico aproximado del desarrollo de **OctoBets**, se han tenido en cuenta el tiempo efectivo dedicado al proyecto, así como los recursos técnicos utilizados y los servicios externos requeridos.

1. Coste en horas de desarrollo

El desarrollo ha sido realizado por una única persona, con una dedicación aproximada de **50 horas** repartidas entre diseño, programación, pruebas, despliegue y documentación.

Aplicando una tarifa orientativa de desarrollador junior de **15 €/h**, el coste estimado sería:

50 h × 15 €/h = 750 €

2. Infraestructura y servicios

Cuadro 8: Costes de infraestructura y servicios utilizados

Concepto	**Coste estimado**
VPS para despliegue (2 meses)	20 €
Dominio web (anual)	10 €
Certificado SSL	0 € (gratuito)
Herramientas de desarrollo (IDE, Figma, etc.)	0 € (versiones gratuitas)
Plataforma de control de versiones (GitHub)	0 € (uso gratuito)
Stripe (modo test)	0 €
Total servicios	**30 € aprox.**

3. Coste total estimado del proyecto

Cuadro 9: Resumen del coste total del proyecto

Categoría	**Coste**
Desarrollo (50 h)	750 €
Servicios externos	30 €
Total	**780 €**

Este presupuesto refleja una estimación orientativa basada en el desarrollo individual. En un contexto profesional, el coste podría incrementarse debido a

factores como contratación de personal, licencias comerciales o soporte técnico especializado.

10 Conclusiones

El desarrollo de **OctoBets** ha supuesto una experiencia completa de creación de una plataforma web funcional, desde su concepción inicial hasta su despliegue final. A pesar de que el proyecto partía con un enfoque más ambicioso que incluía una sección de juegos de casino, el proceso de análisis y priorización permitió redefinir de forma realista el alcance, centrándose en la parte principal: la gestión de apuestas personalizadas entre usuarios.

La plataforma conseguida cumple con los objetivos planteados: permite a cualquier usuario registrarse, explorar apuestas públicas, crear nuevas, participar en ellas, gestionar su cartera y consultar su historial de movimientos. Todo esto se ha desarrollado con tecnologías actuales y robustas, como **Spring Boot**, **Angular** y **Tailwind**, integrando prácticas seguras como **JWT** para la autenticación, y herramientas modernas como **Stripe** para simular pagos.

Durante el desarrollo se han puesto en práctica conceptos clave aprendidos a lo largo del ciclo formativo, como la programación en capas, la separación de responsabilidades, el diseño de base de datos relacional o la importancia de una interfaz clara e intuitiva. Además, el uso de metodologías ágiles como **Scrum** ha facilitado una adaptación continua a los imprevistos y a la evolución del propio proyecto.

Uno de los mayores aprendizajes ha sido precisamente esa capacidad de redefinir el proyecto sin perder su esencia, priorizando la calidad del resultado entregado por encima de intentar abarcar más de lo posible en el tiempo disponible.

En definitiva, **OctoBets** es una plataforma sólida y abierta a futuras ampliaciones, pero también una demostración práctica de todo el conocimiento adquirido durante el ciclo de **Desarrollo de Aplicaciones Web**.