# 

# 

# 

# 

# 

# Plan de Proyecto

Balonix

**Integrantes:**

**Franco Toledo**

**Donkan Álvarez**

**Fabiola Betancourt**

# **Problema**

El problema principal es que los equipos de fútbol amateur carecen de una herramienta que les permita emparejar rivales y poder coordinar de forma ordenada todo el ciclo de un partido: desde el desafío inicial, la negociación de fecha, hora y lugar, hasta la confirmación, la realización del encuentro y la finalización del partido. Hoy este proceso ocurre de manera dispersa en canales informales como WhatsApp o Instagram, lo que dificulta la organización, al no existir reputación, incrementando así el tiempo de coordinación, elevando la tasa de cancelaciones y reduciendo la confianza entre equipos y con los administradores de canchas.

# **Solución propuesta**

La solución que proponemos es una plataforma web y app con enfoque “Matchmaking”, para conectar equipos de fútbol amateur según su ubicación y disponibilidad. Esta herramienta permite que los equipos se coordinen de manera estructurada y organizada, desde el inicio, coordinación de fecha, hora y lugar hasta la confirmación, partido y finalización.

Esta solución también tendrá recordatorios y notificaciones al WhatsApp junto con el registro de resultados e historial de partidos de cada equipo visibles para los demás de forma que un equipo sabrá contra quien és que va a desafiar a un partido, asegurando una comunidad competitiva

También habrá una visibilidad básica de canchas dentro del flujo de organización de un partido, disminuyendo el tiempo invertido en coordinacion y comunicacion

# **Beneficiarios**

Las comunidades deportivas y equipos amateur son los principales beneficiarios de la plataforma. Gracias a ella, podrán encontrar rivales compatibles de manera rápida, evitando largas idas y vueltas en la coordinación. Esto se traduce en una mayor tasa de partidos concretados y en la posibilidad de contar con una reputación visible que genere confianza.

También, los administradores y encargados de canchas se verían favorecidos al contar con mayor visibilidad frente a los equipos y con un canal ordenado de solicitudes, lo que aumenta las probabilidades de concretar reservas y optimizar la ocupación de los recintos, especialmente en horarios con baja demanda. La plataforma, de este modo, soluciona problemas habituales como las consultas dispersas y la dificultad para mantener un flujo constante de reservas.

# **Alcance**

El alcance principal es validar que los equipos de fútbol puedan encontrar rivales compatibles y concretar partidos de manera rápida. De esta manera, se busca comprobar que la plataforma resuelve el problema central de coordinación y confianza entre equipos.

Contemplamos un sistema de autenticación y cuentas con registro e inicio de sesión vía email, junto con verificación básica de correo. Cada equipo contará con un perfil, que incluye información esencial como nombre, comuna o zona, disponibilidad horaria y datos de contacto. También se podrá añadir una plantilla básica de jugadores de cada equipo.

La funcionalidad de buscar posibles rivales permitirá listar equipos filtrables por zona y otros filtros, así como enviar y recibir propuestas iniciales de partido que incluyan fecha, hora y cancha sugerida. El proceso de negociación se simplificará mediante contraofertas hasta llegar a un acuerdo o rechazar el desafío a través de la plataforma y notificaciones por WhatsApp. Una vez acordado, la gestión de partidos se reflejará en estados claros: pendiente, confirmado, en curso, finalizado. Además, se enviarán recordatorios automáticos por WhatsApp/Email.

El sistema también permitirá al momento de finalizar el partido el registro de resultados, incluyendo marcador final y condición del partido (ganado, empatado o perdido, que se determinará automáticamente según el resultado) lo que se verá reflejado en el perfil del equipo. Tras cada encuentro, los equipos se calificarán mutuamente mediante una puntuación simple de 1 a 5 y etiquetas asociadas a fair play y puntualidad. Con esto se generará un puntaje de reputación visible en cada perfil de equipo.

**Metodología**

En el contexto de este proyecto se utilizará una metodología tradicional de desarrollo en cascada, la cual permite estructurar el trabajo en fases consecutivas, asegurando un orden claro y la documentación en cada etapa. Dado que el equipo es reducido y se cuenta con un tiempo limitado, este enfoque facilita la planificación lineal y la entrega dentro de los plazos definidos.

Las fases consideradas son:

* Análisis y levantamiento de requerimientos  
   Se definen y documentan los requerimientos funcionales y no funcionales del sistema, tomando en cuenta las necesidades de los equipos de fútbol y las canchas.
* Diseño  
   Se elaboran los diagramas de clases con sus métodos, casos de uso y el diseño de la arquitectura general. Además, se definen los flujos de usuario y la interfaz básica de la aplicación.
* Desarrollo  
   Se implementan los módulos principales en orden: autenticación, perfiles de equipo, gestión de desafíos, reputación y notificaciones. Cada módulo se construye según lo especificado en la fase de diseño.
* Pruebas  
   Se realizan pruebas unitarias, de integración y funcionalidades para asegurar que el sistema cumpla con los requerimientos
* Implementación  
   El sistema se despliega en la infraestructura definida (Firebase y servicios asociados), asegurando la disponibilidad y el correcto funcionamiento de los módulos principales.
* Mantenimiento  
   Una vez en producción, se contemplan ajustes menores, corrección de errores y mejoras basadas en la retroalimentación de los usuarios.

# **Requerimientos**

## **Requerimientos Funcionales (RF)**

El sistema debe otorgar funcionalidades que permitan a los usuarios coordinar partidos de manera eficiente y transparente. Los principales requerimientos funcionales son:

* RF-01 Autenticación: los usuarios pueden registrarse e iniciar sesión usando email/contraseña, con verificación obligatoria de correo antes de crear o unirse a un equipo.
* RF-02 Gestión de perfil de equipo: creación y edición de perfil con información como nombre del equipo, comuna y región, disponibilidad horaria y medio de contacto, además de subir un logo o foto.
* RF-03 Plantilla de jugadores: registro de datos de jugadores por cada equipo
* RF-04 Descubrimiento de equipos: listado de equipos con filtros por zona, fair-play o puntualidad.
* RF-05 Creación de desafío: envío de desafío a otro equipo con fecha, hora, comuna y cancha sugerida.
* RF-06 Recepción y respuesta de desafío: aceptación, rechazo o contraoferta por parte del equipo receptor, permitiendo editar fecha, hora y/o lugar.
* RF-07 Negociación y confirmación: flujo de estados: pendiente de aceptación, confirmado, en curso, finalizado.
* RF-08 Panel de próximos partidos e historial de ya agendados: visualización de partidos confirmados y registros históricos con estados y resultados.
* RF-09 Recordatorios automáticos: recordatorios de partidos por aceptar y por comenzar via email/whatsapp
* RF-10 Registro de resultado: ingreso de marcador por ambos equipos; desacuerdos quedan en estado “pendiente” hasta que ambos coincidan.
* RF-11 Calificación cruzada: puntuación de 1 a 5 con tags de fair play y puntualidad, generando reputación visible.
* RF-12 Visibilidad de canchas: listado básico de canchas con nombre y dirección al momento de desafiar a otro equipo
* RF-13 Notificaciones de sistema: avisos sobre nuevos desafíos, contraofertas y confirmaciones.
* RF-14 Roles y permisos: roles “capitán” y “jugador” de cada equipo teniendo el capitán más permisos sobre el equipo que el jugador
* RF-15 Registro de eventos clave: registro de eventos importantes en el sistema (creación, confirmación, cancelación, finalización) con fecha, hora, lugar,etc.
* RF-16 Soporte móvil y web: diseño responsivo, compartido entre Ionic App y web Angular.

## **Requerimientos No Funcionales (RNF)**

Los requerimientos no funcionales aseguran que el sistema cumpla con estándares de rendimiento, seguridad, escalabilidad y usabilidad:

* RNF-01 Rendimiento: tiempo de respuesta en páginas principales de 2.5 s , con listados paginados o carga incremental.
* RNF-02 Disponibilidad: 99% en horario hábil, deploy sin caídas prolongadas en Functions o Hosting.
* RNF-03 Seguridad: reglas de Firestore basadas en rol/propietario, validación de entradas en cliente y Functions.
* RNF-04 Privacidad y cumplimiento: políticas visibles de privacidad, términos y verificación de emails.
* RNF-05 Observabilidad: logs estructurados en Cloud Functions
* RNF-06 Compatibilidad: web responsive (Angular) y app híbrida (Ionic) para Android; iOS.
* RNF-07 Accesibilidad: contraste mínimo y etiquetas en formularios.
* RNF-08 Escalabilidad inicial: colecciones indexadas para filtros frecuentes (zona, nivel) y control de costos en lecturas.

## 

## 

## 

# 

# **Diagrama de clase**

# **Casos de Uso**

# 

# **Diagrama de base de datos**

# **Tecnologías Propuestas**

## **Frontend (Web y Móvil)**

* Angular: Framework principal para la web, cumpliendo RNF-06 (compatibilidad web responsive) y RNF-07 (accesibilidad).
* Ionic Framework: Desarrollo híbrido para Android/iOS y soporte PWA (RF-16, RNF-06).
* Capacitor: Integración de funcionalidades nativas en la app móvil.
* Tailwind CSS: Estilos rápidos, consistentes y accesibles.
* Backend y Lógica Firebase Cloud Functions (Node.js): Procesamiento backend serverless, ideal para autenticación, notificaciones, recordatorios automáticos y lógica de negocio (RF-09, RF-13, RF-15).

## **Base de Datos y Almacenamiento**

* Firebase Firestore: Base de datos NoSQL escalable, con soporte a colecciones indexadas para filtros por zona y fair play (RF-04, RNF-08).
* Firebase Storage: Almacenamiento de imágenes (logos, fotos de equipos) — RF-02.

## **Autenticación y Seguridad**

* Firebase Authentication: Soporte para email/contraseña, Google y verificación de correo (RF-01, RNF-03).
* Firestore Security Rules: Control de acceso basado en roles (capitán/jugador) — RF-14, RNF-03.

## **Notificaciones y Comunicación**

* Resend / Manychat: Recordatorios por email y WhatsApp (RF-09).

## **Infraestructura**

* Firebase Hosting: Hosting para web (RNF-02).
* GitHub Actions / GitLab CI: CI/CD con despliegues automatizados sin caídas (RNF-02).

## **Monitoreo y Observabilidad**

* Firebase Crashlytics + Google Cloud Logging: Manejo de errores, logs y métricas clave (RNF-05).
* Google Analytics / Mixpanel: Métricas de uso y retención de usuarios.

## **Privacidad y Cumplimiento**

* Políticas visibles en frontend: RNF-04 (términos y condiciones, privacidad).

# **Cronograma de Desarrollo (4 Semanas)**

**Semana 1: Descubrimiento – Requerimientos**

* **A1:** Levantamiento rápido de requerimientos con equipos y canchas.

**Semana 2: Diseño – Flujo de Usuario**

* **A2:** Definición de flujo de partidos, roles y reputación.

**Semana 3: Diseño – Prototipos**

* **A3:** Diseño de prototipos de baja fidelidad para validación rápida.

**Semana 4: Desarrollo Core – Autenticación y Perfiles**

* **A4:** Implementación de autenticación y perfiles de equipos (RF-01, RF-02, RF-14).

**Semana 5: Desarrollo Core – Jugadores y Descubrimiento**

* **A5:** Registro básico de jugadores (RF-03).
* **A6:** Descubrimiento de equipos (RF-04).

**Semana 6: Desarrollo Core – Desafíos**

* **A7:** Creación y recepción de desafíos (RF-05, RF-06).
* **A8:** Flujo de aceptación y confirmación (RF-07).

**Semana 7: Funcionalidades de Coordinación – Partidos e Historial**

* **A9:** Registro de partidos y resultados (RF-08, RF-10).
* **A10:** Historial y panel de próximos partidos.

**Semana 8: Funcionalidades de Coordinación – Reputación**

* **A11:** Calificación cruzada de equipos (RF-11).
* **A12:** Sistema de fair play y puntualidad.

**Semana 9: Funcionalidades de Coordinación – Notificaciones**

* **A13:** Recordatorios automáticos de partidos (RF-09).
* **A14:** Notificaciones de sistema (RF-13).

**Semana 10: Ajustes Finales**

* **A15:** Visibilidad básica de canchas y selección en el flujo de desafío (RF-12).
* **A16:** Corrección de errores y mejoras de UI/UX.

**AÑADIR HITOS DE CONTROL (PRESENTACIONES DEL RAMO)**

**Semana 11: Pruebas, Piloto y Cierre**

* **A17:** Pruebas funcionales de todos los módulos.
* **A18:** Prueba piloto rápida (incluye un despliegue inicial).
* **A19:** Recopilación de métricas iniciales de coordinación y cancelaciones.

# **Carta Gantt**

# **Costos**

### **Hardware** El proyecto utilizará hardware propio (computadores personales de los integrantes), por lo que no se generarán costos adicionales asociados a la compra de equipos.

### **Firebase**

### Firestore: lecturas y escrituras bajas (menos de 50,000/mes). Costo: $0 USD, entra dentro del plan gratuito.

### Cloud Functions: pocas ejecuciones diarias (recordatorios, notificaciones). Costo: $0 USD, plan gratuito cubre hasta 2 millones de invocaciones.

### Hosting: tráfico web muy bajo, sitio ligero. Costo: $0 USD, plan gratuito cubre 1 GB almacenamiento y 10 GB transferencia.

### Storage: solo algunos logos y fotos, < 1 GB. Costo: $0 USD, plan gratuito cubre hasta 5 GB.

### Total Firebase: ~$0 USD/mes para prototipo, sin embargo esto depende del uso. Con un mayor volumen de usuarios, el costo puede aumentar hasta ~$360 USD/mes.

### **Resend (emails)** Envío limitado de recordatorios por email (<3,000/mes). Costo: $0 USD (plan gratuito). En caso de mayor demanda, podría aumentar hasta $50 USD/mes.

### **ManyChat (WhatsApp)** Comunicación limitada a <1,000 contactos. Costo: $15 USD (plan pro).

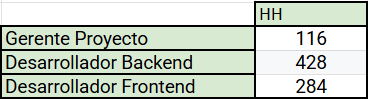
### **Dominio**

Compra del dominio para el uso en el hosting y Resend.

Costo: $10.27 USD/Año

1. Recursos Humanos

Gerente de proyecto: Horas totales 116, CLP $7100 por hora, son CLP $823.000



# **Riesgos Principales**

## **1. Riesgos Técnicos**

* Integración Resend y ManyChat: Posibles errores en envíos masivos de emails o WhatsApp.
* Retrasos o duplicación de notificaciones.
* Compatibilidad móvil/web: Problemas de visualización o funcionalidades inconsistentes entre la app Ionic y la web Angular, especialmente en dispositivos con diferentes tamaños de pantalla o versiones de Android.
* Seguridad y permisos: Implementación incorrecta de Firestore Security Rules podría exponer datos de usuarios o permitir acciones no autorizadas.

## **2. Riesgos de Gestión y Alcance**

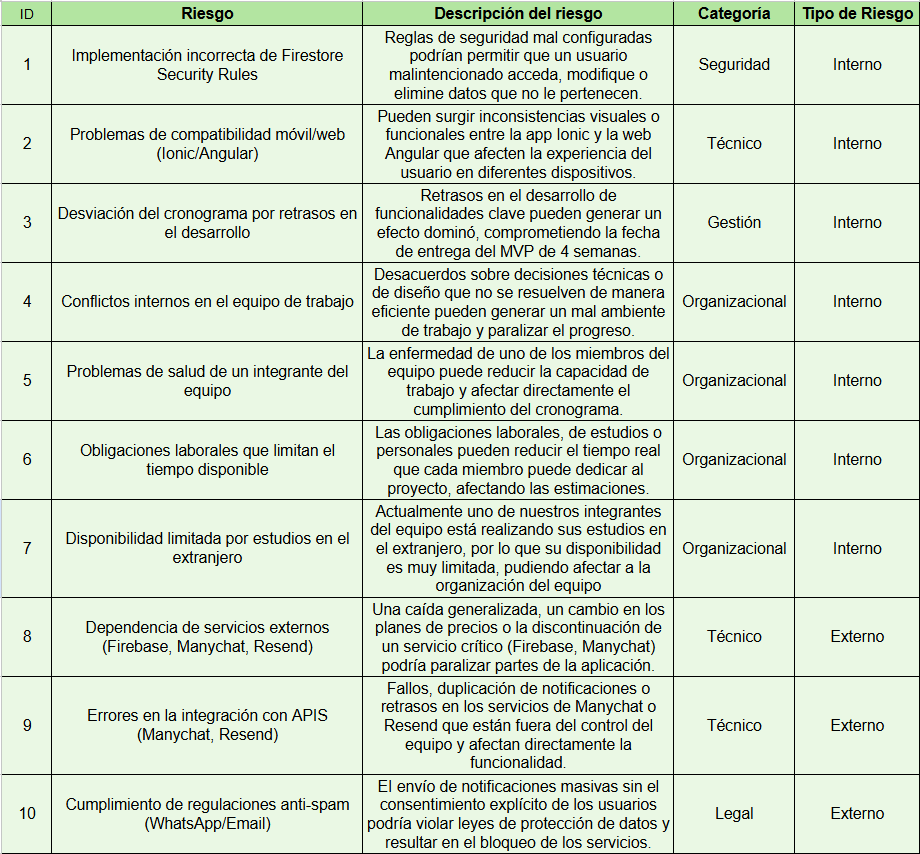
* Desviación del cronograma: Retrasos en el desarrollo de core features (autenticación, perfiles, desafíos) pueden afectar el desarrollo del proyecto.
* Dependencia de servicios externos: Resend, ManyChat y Firebase son críticos; cualquier interrupción o cambio en sus planes de precios impactaría directamente la operación.

## **3. Riesgos Legales y de Privacidad**

* Mensajes automáticos: Uso de WhatsApp y email debe cumplir regulaciones anti-spam; envíos masivos sin consentimiento pueden generar bloqueos o sanciones.

## **4. Riesgos Organizacionales**

* Conflictos de equipo: el equipo de trabajo puede presentar conflictos u ideas en las que no estén de acuerdo con los demás e interferir con el cronograma del proyecto
* Problemas de salud: posibilidad de que un integrante o varios del equipo se enferme y afecte al cronograma
* Obligaciones laborales de los integrantes: la disposición del tiempo de cada integrante está limitado debido a que todos debemos realizar la practica laboral, lo cual puede afectar también al tiempo de desarrollo del proyecto
* Disponibilidad de integrantes por estudios: actualmente uno de nuestros integrantes del equipo está realizando sus estudios en el extranjero, por lo que su disponibilidad es muy limitada, pudiendo afectar a la organización del equipo



# **Pruebas**

## **1. Pruebas funcionales**

Se validará que las funcionalidades descritas en los requerimientos estén operativas y cumplan con lo esperado desde la perspectiva del usuario, sin analizar el código:

* Registro e inicio de sesión de equipos y jugadores.
* Creación, envío y aceptación de desafíos entre equipos.
* Registro y visualización de partidos jugados.
* Calificación cruzada de equipos y jugadores.
* Notificaciones automáticas y recordatorios de partidos.

## **2. Pruebas de Integración**

* Flujo Firebase Auth + Firestore + Roles: autenticación y restricción correcta.
* Resend (emails): entrega correcta en inbox, sin ir a spam.
* ManyChat (WhatsApp): mensajes llegan en el tiempo esperado, no duplicados.
* Capacitor (móvil): acceso a cámara/archivos para subir logos.

## **3. Pruebas de Usabilidad y Experiencia de Usuario**

* Probar el flujo completo (crear equipo, desafiar, negociar, confirmar, jugar, registrar resultado, calificación).
* Evaluar si un usuario sin tutorial entiende el flujo.
* Validar claridad de mensajes de error (ej. “fecha inválida” vs “error 400”).