

**fullstack 2**

SIGLAS Y SECCIÓN (Ejemplo: DRY7112 - 002D)

NOMBRE DE LA SEDE

**“realms in discord”**

[ISAAC GABRIEL RIVAS VARELA](mailto:i.rivas@duocuc.cl)

Prof. nombre y apellido profesor/a

12/11/2025

**ÍNDICE DE CONTENIDO**

[**introducción 3**](#_fmwri0lpsgtx)

[**contexto general del proyecto 3**](#_xcetpxlfabxn)

[**necesidades y expectativas 4**](#_o563vbxvvx89)

[**Herramientas y tecnologías 5**](#_d6uab7irrlop)

# introducción

A continuación se presentan las ERS (especificaciones de requisito del sistema) relacionado al proyecto “REALMS IN DISCORD”, de esta forma definiendo de forma clara las herramientas,necesidades,objetivos,restricciones y funcionalidad.

# contexto general del proyecto

Principalmente se dará a conocer la idea general de que tratará el proyecto para entender de mejor manera el proyecto con tal de hacer posteriormente referencia explícita de las necesidades en términos de herramientas y técnicas de desarrollo software.

Idea principal → TCG.

Temáticas → Juego de estrategia táctica y construcción de mazo.

Pitch → “Imagina un juego de cartas donde no gana siempre el que tiene el mazo más caro o poderoso, sino el que sabe tomar las mejores decisiones en el momento correcto. Ese es nuestro TCG: un proyecto de programación que combina estrategia táctica y construcción de mazo de manera equilibrada. Actualmente contamos con dos facciones iniciales: los **Caballeros Solares**, defensores del orden, y **La Corrupción**, una fuerza caótica que evoluciona durante la partida. Nuestro objetivo es ofrecer una experiencia justa, competitiva y divertida, donde la habilidad y la estrategia importan más que la suerte o la inversión. Estamos buscando apoyo y retroalimentación para seguir expandiendo este universo.”

# necesidades y expectativas

**Requisito funcional**

1. CRUD de la información de un usuario
2. accesibilidad multiplataforma (pc, movil)
3. biblioteca accesible del contenido (cartas)
4. chat con amigos
5. sistema de buzón de amistades
6. sistema de reporte de errores
7. sistema de reporte de usuarios
8. conectividad online

**Requisito no funcional**

1. rendimiento estable
2. usabilidad
3. seguridad
4. mantenibilidad
5. interfaz de usuario simple
6. atributos de calidad

**Regla de negocio**

Las reglas de negocios que se definen en el proyecto de “REALMS IN DISORD” se pueden definir en conceptos clave para una correcta transparencia del objetivo del mismo.

* **contenido continuo:** se espera que los usuarios reciban contenido mediante actualizaciones, parches y cambios de balance con tal de mejorar la experiencia del usuario y mantener un sentimiento de preocupación por parte de los desarrolladores.
* **datos protegidos:** el sistema almacena datos sensibles en el sistema por lo tanto estos serán protegidos.
* **respuesta rápida de errores:** se buscará el medio lo más rápido y óptimo posible para la resolución de errores informáticos que puedan arruinar la experiencia del usuario

# Herramientas y tecnologías

Se especificarán las herramientas tecnológicas que nos permitirán como equipo de desarrollo realizar el proyecto.

# 1. Front-End (Interfaz del juego)

- HTML/CSS/JavaScript: base obligatoria para la parte visual.  
- Framework sugerido:  
 • React.js: muy bueno para interfaces interactivas como tableros, drag & drop de cartas, estados dinámicos.  
 • Alternativa: Vue.js, más simple de aprender.  
- Librerías útiles:  
 • PixiJS: para animaciones y efectos visuales de cartas.  
 • Phaser.js: si se busca un enfoque de motor de juego ligero.

# 2. Back-End (Lógica y microservicios)

- Kotlin como lenguaje principal.  
- Frameworks:  
 • Ktor: ideal para microservicios, rápido y ligero.  
 • Spring Boot con Kotlin: más robusto, estándar en la industria, con soporte para microservicios.  
- Recomendación:  
 • Ktor si se busca ligereza.  
 • Spring Boot si se quiere un enfoque académico y completo.

# 3. Comunicación cliente-servidor

- REST API: sencillo, cada microservicio expone endpoints.  
- WebSockets: fundamental en juegos multijugador para tiempo real.  
 • Tanto Ktor como Spring Boot soportan WebSockets.  
- JSON: formato recomendado para intercambio de datos.

# 4. Base de datos

- MongoDB: Para persistencia de usuarios, progreso y mazos.  
- Redis (opcional): para cachear estados de partidas en tiempo real. Además reduce la carga en MongoDB

# 5. Arquitectura de microservicios

- División en servicios pequeños:  
 • Auth Service: login y registro de jugadores.  
 • Deck Service: gestión de cartas y mazos.  
 • Match Service: lógica y reglas de las partidas.  
 • Notification Service (opcional): chat o mensajes.  
- Comunicación entre microservicios:  
 • REST interno o gRPC (más eficiente).

# 6. Android Studio (exportación)

- Puede envolver el front-end web en una WebView.  
- Alternativamente, se puede hacer una app nativa en Kotlin que consuma los microservicios vía API.  
- Integración recomendada: Kotlin (Android) como app cliente + backend en Kotlin (Ktor/Spring Boot).

# Resumen recomendado

- Front-End: React.js + PixiJS (con JavaScript).  
- Back-End: además de realizar un levantamiento en la nube de la base de datos para que tanto el back como el front estén conectados de forma simultánea Spring Boot (completo).  
- Comunicación: WebSockets + REST.  
- Base de dados: MongoDB  
- App final: Android Studio con Kotlin (consumiendo APIs o WebView).

# enfoque de proyecto

El enfoque principal del proyecto es buscar que el usuario se sienta cómodo con el producto digital que estamos ofreciendo, a través del mismo juego, ilustraciones y formato competitivo, por esto la preocupación principal es buscar que la experiencia de juego sea la más adecuada y balanceada en ámbito competitivo y estratégico.

La estrategia de desarrollo a ocupar para un desarrollo óptimo del proyecto será el Sprint y daily scrum para mantener una organización del desarrollo optimizada y junto al daily scrum para la comunicación constante permitiendo una flexibilidad y atencion a los detalle simportante a la hora de asignar tareas.