Acta de Reunión: Sprint 4 (Reunión 2)

Información General

```
Proyecto: Juego de Cartas "Ronda Marroqui"
Tipo de reunión: Seguimiento de Implementación y Definición Técnica
Fecha y hora: 01/12/2025
Duración: 2 Horas 30 Minutos
Lugar: Presencial - Campus UPV
Asistentes:
Yahya Aboulafiya
Adrián Hoyos Sánchez
Souhail Batah
Carlos Robledo Badía
```

1. Objetivo de la Reunión

Revisar lo que avanzamos desde la última reunión, definir bien cómo vamos a implementar los WebSockets, y hacer un plan de trabajo para terminar el Sprint 4.

2. Temas Tratados

2.1. Revisión del Progreso

Las tareas de la reunión pasada se completaron:

Investigación Técnica Completada: Revisamos la documentación de Socket.io y confirmamos que es la mejor opción porque es fácil de usar y tiene fallback automático a polling.

Análisis del Código Actual: Revisamos bien el código del Sprint 3 para ver dónde tenemos que meter los WebSockets, sobre todo en server.js, index.html y sala-espera.html.

Optimizaciones Previas: Optimizamos algunas queries SQL del sistema de torneos, agregando índices para que vaya más rápido antes de meter el tiempo real.

2.2. Arquitectura de WebSockets

Definimos cómo va a funcionar técnicamente:

Biblioteca a Utilizar: Socket.io versión 4.x para servidor y cliente.

Sistema de Salas (Rooms): Cada partida es una sala independiente (`partida_{id_partida}`), así los mensajes solo llegan a los jugadores de esa partida.

Eventos Principales: - `connection`: Cuando un cliente se conecta al servidor - `join_partida`: Jugador se une a una sala de partida - `movimiento`: Jugador realiza un movimiento - `estado_actualizado`: Servidor notifica cambio de estado - `jugador_unido`: Notifica que un nuevo jugador se unió - `partida_iniciada`: Notifica el inicio de la partida - `disconnect`: Manejo de desconexiones

Flujo de Datos: El servidor tiene la verdad absoluta. Todos los movimientos se validan en el servidor y después se envía el nuevo estado a todos los clientes de la sala.

2.3. Cambios Necesarios en el Código

Lo que hay que modificar:

Backend (server.js): - Importar y configurar Socket.io - Crear el manejador de eventos de socket - Modificar los endpoints de movimiento para que emitan eventos en lugar de solo responder HTTP - Mantener la compatibilidad con la API REST existente durante la transición

Frontend (index.html y sala-espera.html): - Agregar la librería cliente de Socket.io - Eliminar el sistema de polling (setInterval) - Implementar los listeners de eventos de socket - Mantener la lógica de actualización de interfaz

Base de Datos: - No requiere cambios estructurales - El sistema actual de persistencia con JSON seguirá funcionando igual

2.4. Plan de Implementación por Fases

Se acordó un plan de desarrollo incremental:

Fase 1 - Configuración Básica (2-3 días): Instalar socket.io en el proyecto Configurar servidor con soporte de WebSockets Implementar conexión básica cliente-servidor Probar que la conexión funciona correctamente

Fase 2 - Implementación de Eventos (3-4 días): Implementar sistema de salas (rooms) Crear manejadores de eventos en servidor Implementar listeners en cliente Integrar con la lógica existente del juego

Fase 3 - Eliminación de Polling (1-2 días): Remover código de polling del frontend Verificar que toda la sincronización funciona por WebSockets Ajustar tiempos de respuesta y UX

Fase 4 - Testing y Refinamiento (2-3 días): Pruebas con múltiples jugadores simultáneos Pruebas de desconexión y reconexión Manejo de casos extremos Optimización de rendimiento

Fase 5 - Documentación (1-2 días): Actualizar documentación técnica Crear diagramas de flujo de eventos Documentar la API de WebSockets

2.5. Consideraciones de Seguridad y Rendimiento

Cosas importantes que discutimos:

Seguridad: Vamos a mantener la validación de sesión con express-session antes de permitir conexiones. Todo se sigue validando en el servidor.

Rendimiento: Esperamos bajar la latencia de 3 segundos (polling actual) a menos de 100ms con WebSockets.

Escalabilidad: El sistema de rooms nos va a permitir tener varias partidas simultáneas sin que se mezclen. De momento ponemos límite de 10 partidas simultáneas para ver cómo va.

Manejo de Errores: Si alguien se desconecta, intentamos reconectar automáticamente y le mostramos mensajes al usuario.

3. Acuerdos y Asignación de Tareas

Quien hace qué:

Carlos: Configurar Socket.io en el servidor, hacer el sistema de rooms y los manejadores de eventos del backend.

Souhail: Meter Socket.io en el frontend, quitar el polling y hacer las notificaciones visuales.

Adrián: Coordinar la integración entre frontend y backend, testing del sistema completo y manejar las desconexiones.

Yahya: Verificar que todo funciona bien con la base de datos, monitorear el rendimiento y hacer la documentación técnica.

4. Próxima Reunión

Quedar el 8/12/2025 para ver cómo quedó todo, hacer demos y preparar la entrega final del Sprint 4.

Firma de conformidad: Equipo de Desarrollo Ronda Marroquí