MUSICNET

Descripción del diseño del software
26/05/2025

V.3

Valentina Rozo González
Paula Valentina López Cubillos

Historial de Cambios

Versión	Fecha	Sección del documento modificada	Descripción de cambios (corta)	Responsable (S)
V.1	*24/10/2024.*	*Creación del documento*	Se creo el documentó guiándonos de la plantilla suministrada, y se empezó el trabajo en la sección introductoria	*Wilson Sanchez*
V.2	*09/02/2025*	*Arquitectura*	Se desarrollo la sección de arquitectura del sistema donde se realizaron los diferentes diagramas	*Valentina Rozo Valentina López*
V.2	*19/05/2025*	*Arquitectura final*	Se ajustaron todos los diagramas a la arquitectura finalmente desarrollada	*Valentina Rozo Valentina Lopez*

Tabla 1: Historial de cambios

Table of Contents

Introducción

Este documento ofrece una guía exhaustiva para que los desarrolladores comprendan la arquitectura y el diseño detallado de un sistema, cubriendo tanto sus componentes lógicos y físicos como los principales procesos de interacción con el usuario. A través de diagramas y explicaciones, el documento proporciona una vista completa de la estructura y el comportamiento del sistema, facilitando una comprensión clara y práctica para la implementación de cada componente, desde la interfaz de usuario hasta la persistencia de datos.

La audiencia ideal para este documento son desarrolladores y arquitectos de software que buscan una base sólida para implementar o ampliar el sistema descrito. Cada sección está diseñada para brindar una comprensión estructurada de los componentes y sus interacciones, proporcionando herramientas visuales y directrices claras que serán especialmente útiles para aquellos involucrados en la configuración, mantenimiento y mejora de sistemas complejos.

A medida que el lector avance, encontrará información detallada y recursos específicos, organizados para facilitar la referencia rápida y efectiva. Este enfoque permite que tanto nuevos integrantes de un equipo como desarrolladores experimentados puedan beneficiarse al comprender cómo se integran los distintos elementos del sistema, permitiéndoles tomar decisiones informadas en futuras fases de desarrollo o mantenimiento.

Arquitectura

1.1. Vista Lógica del Sistema

Descripción General del Diagrama

El diagrama de componentes para MusicNet describe la relación entre los principales módulos del sistema, sus interfaces, bibliotecas externas y dependencias necesarias para su ejecución.

Componentes Clave para el Diagrama de Componentes

Cliente Web:

- UI desarrollada en HTML/CSS/JS con Phaser.
- PitchDetector con AubioJS.
- Módulo de Visualización (escena y avatares).
- Controlador WebRTC (conexión P2P y canal de datos).
- Comunicación con API PARCNet.

Servidor de Señalización (Node.js):

- Gestión de salas.
- Comunicación WebRTC vía Socket.IO.
- Promoción automática del host.

PARCNet (Python):

- Se encarga de la reconstrucción de audio perdido utilizando un modelo de predicción híbrido basado en técnicas autoregresivas y redes neuronales convolucionales. Este módulo opera dentro de un servidor WebRTC y recibe audio a través de un canal de medios (track) desde el navegador. La nota detectada se devuelve mediante un canal RTCDataChannel, todo en tiempo real.

Dependencias externas:

- Phaser.js, WebRTC, WebSocket, AubioJS.

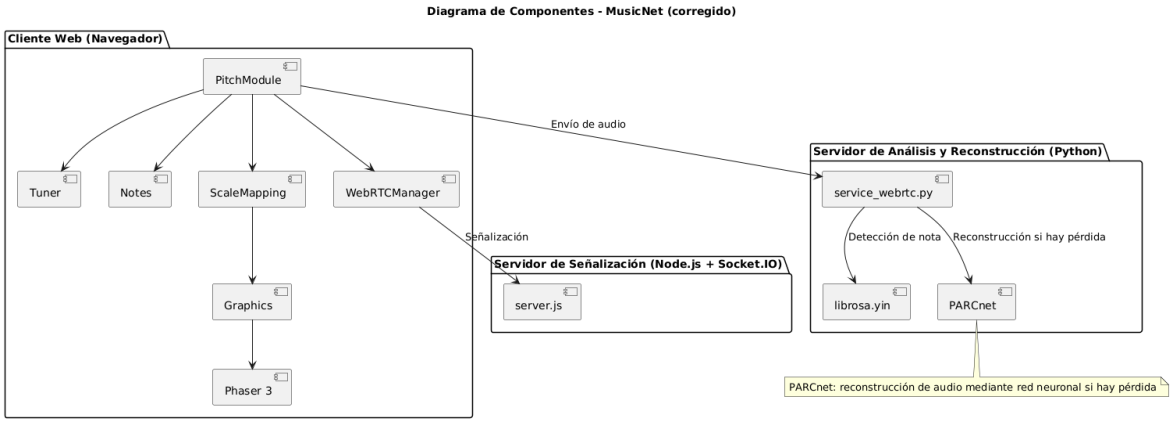

Ilustración 1 Diagrama de Componentes

1.2. Vista Física del Sistema

Descripción General del Diagrama

Este diagrama muestra cómo los componentes del sistema están instalados y ejecutados en diferentes nodos, enfatizando los lugares donde residen las bibliotecas y componentes del juego.

Componentes Clave para el Diagrama de Despliegue

La arquitectura distribuida híbrida consta de:

- Navegador del Usuario (Cliente Web):

- Ejecuta UI, detección de tono, renderizado y lógica de juego.
- Establece conexión directa P2P con otro jugador vía WebRTC.
- Requiere acceso a micrófono y red TCP/UDP.

- Servidor de Señalización (Render.com):
  - Desplegado como servicio en Node.js.
  - Canal WebSocket para negociación de WebRTC.
- Servidor de Compensación de Audio (Render.com):
  - Ejecuta modelo PARCNet (Python).
  - Recibe y responde solicitudes HTTP con audio reconstruido.

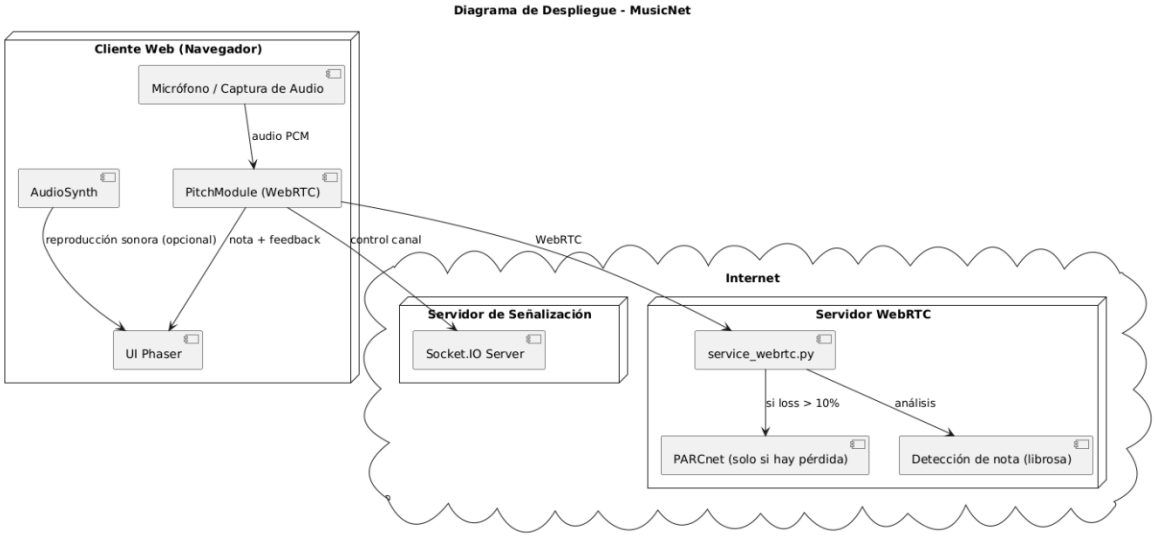

Ilustración 2 Diagrama de despliegue

1.3. Vista de Procesos del Sistema

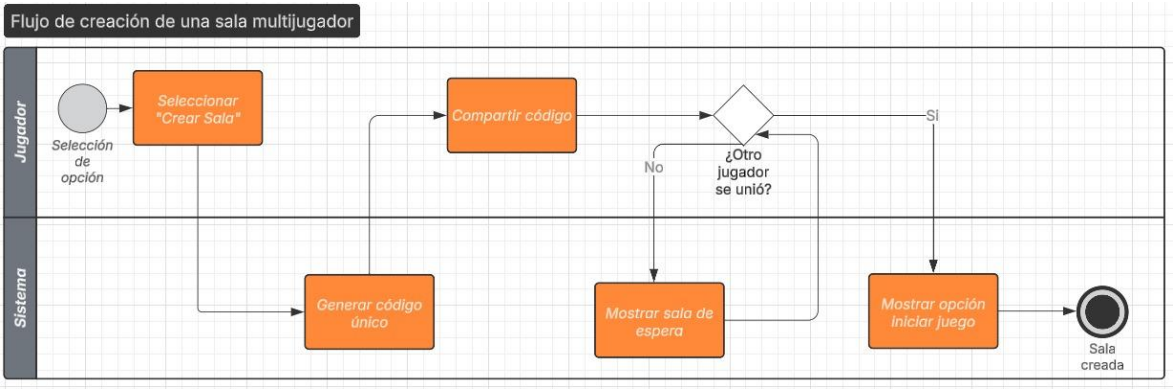

Ilustración 3. Creación sala multijugador

Descripción General Creación de una Sala Multijugador:

Este flujo describe el proceso mediante el cual un jugador crea una sala en el modo multijugador, permitiendo que otro jugador se una mediante un código único generado por el sistema.

Explicación del flujo:

1. El jugador selecciona la opción "Crear Sala".
2. El sistema genera un código único y lo muestra al jugador.
3. El jugador comparte el código con otro usuario.

4. Se verifica si otro jugador ha ingresado el código:
    a. Sí: Se muestra la opción de iniciar la partida.
    b. No: Se mantiene la sala en espera.
5. Una vez que ambos jugadores están conectados, se puede iniciar la partida.

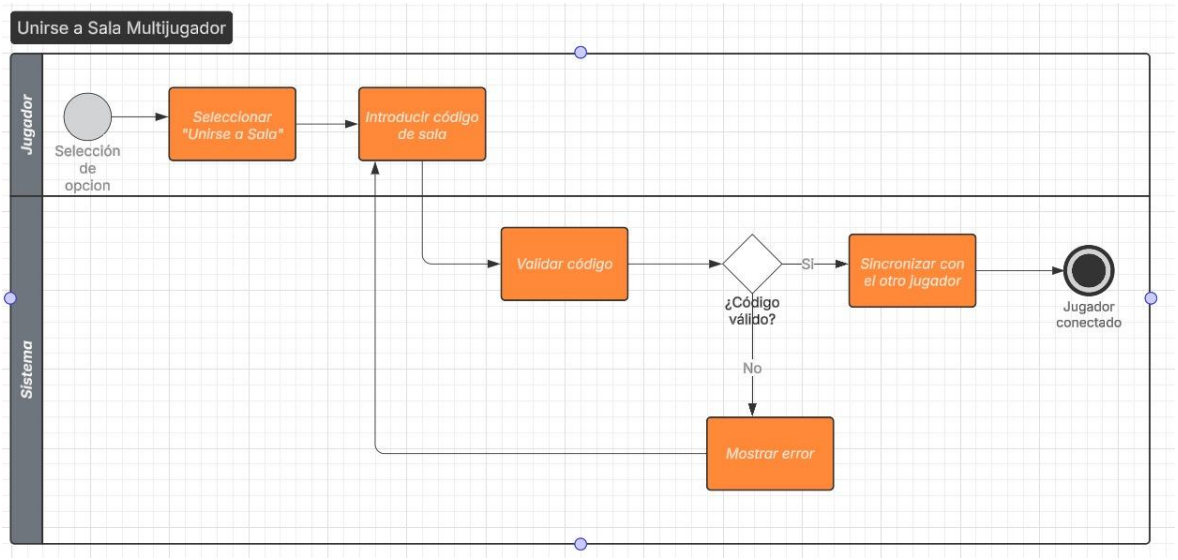

Ilustración 4.Unirse sala multijugador

Descripción general Unirse a una Sala Multijugador:

Este flujo describe el proceso mediante el cual un jugador ingresa a una sala existente en el modo multijugador utilizando un código único.

Explicación del flujo:

1. El jugador selecciona "Unirse a Sala".
2. Introduce el código único de la sala.
3. El sistema valida el código ingresado:
    a. Código válido: Se conecta al otro jugador y se muestra la interfaz del juego.
    b. Código inválido: Se muestra un mensaje de error.
4. Si la conexión es exitosa, los jugadores pueden iniciar la partida.

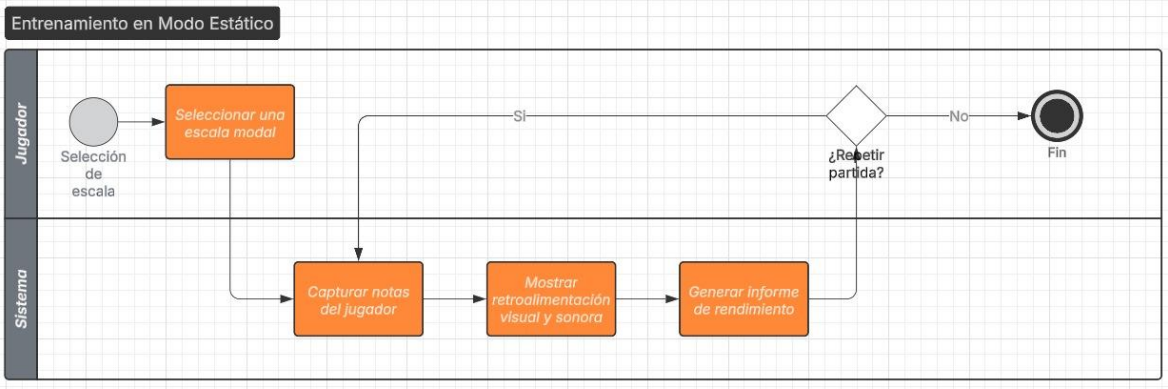

Ilustración 5. Modo Estático

Descripción General Entrenamiento En Modo Estático:

Este flujo describe el proceso de entrenamiento del jugador en un modo donde debe tocar una escala modal y recibir retroalimentación sobre su ejecución.

Explicación del flujo:

1. El jugador selecciona una escala para entrenar.
2. El sistema captura las notas tocadas.
3. Se evalúa la precisión de la ejecución.
4. El sistema muestra retroalimentación visual y sonora.
5. Se genera un informe con estadísticas del desempeño.
6. El jugador decide si repetir la sesión o finalizar el entrenamiento.

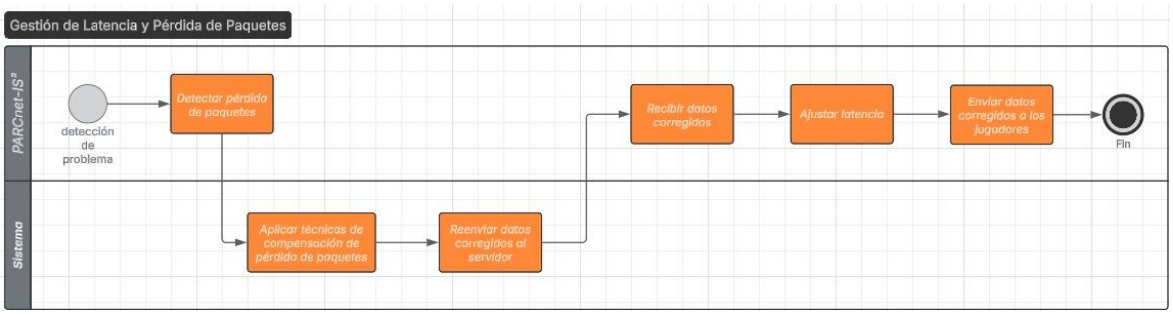

Ilustración 6. Gestión de Latencia

Descripción general Gestión de Latencia:

Este flujo describe el proceso mediante el cual el sistema maneja la pérdida de paquetes de audio en la comunicación en tiempo real para garantizar una experiencia fluida.

Explicación del flujo:

1. El sistema PARCnet-IS² detecta pérdida de paquetes.
2. Se analiza si los datos perdidos pueden recuperarse
3. Los datos corregidos se envían al servidor.
4. El sistema ajusta la latencia en tiempo real para evitar desincronización.
5. La información corregida se transmite a los jugadores.

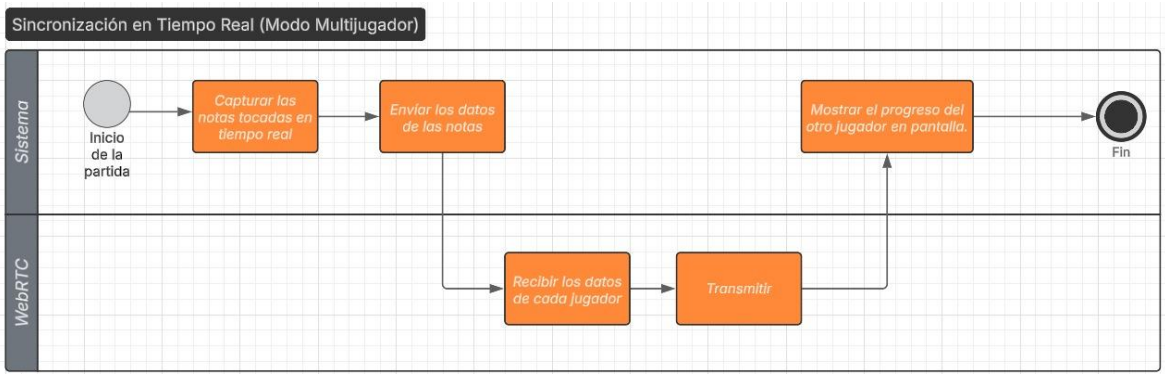

Ilustración 7. Sincronización en Tiempo Real

Descripción general Sincronización en Tiempo Real (Modo Multijugador):

Este flujo muestra cómo el sistema maneja la sincronización de notas en tiempo real en el modo multijugador mediante WebRTC.

Explicación del flujo:

1. El sistema captura las notas que toca cada jugador.

2. Los datos son enviados a WebRTC.
3. WebRTC transmite la información entre los jugadores.
4. El sistema muestra en pantalla el progreso de ambos jugadores.
5. Si se detecta desincronización, se ajusta la latencia automáticamente.

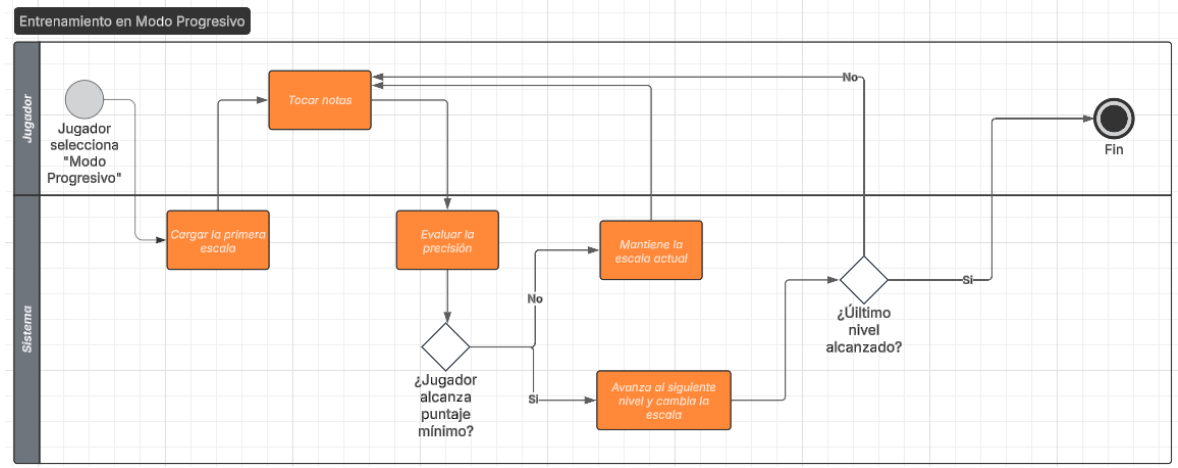

Ilustración 8 Modo Progresivo

Descripción general Juego Modo Progresivo:

Este flujo describe el modo de entrenamiento donde el jugador avanza de nivel si su ejecución es correcta, aumentando la dificultad progresivamente.

Explicación del flujo:

1. El jugador selecciona el modo progresivo.
2. El sistema carga una escala inicial.
3. Se captura la ejecución del jugador y se evalúa su precisión.
4. Si el jugador obtiene el puntaje mínimo:
    a. Sí: Avanza al siguiente nivel con una escala más difícil.
    b. No: Repite el nivel actual.
5. Si se alcanza el último nivel o el jugador decide retirarse, el modo finaliza.

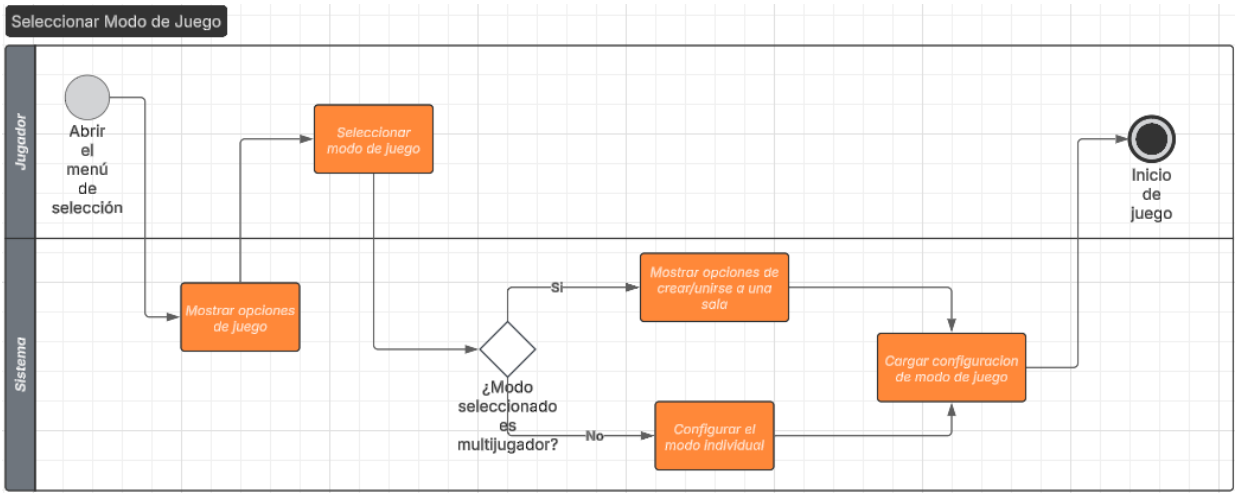

Ilustración 9 Seleccionar Modo de Juego

Descripción general Seleccionar Modo de Juego:

Este flujo muestra el proceso de selección del modo de juego, permitiendo al jugador elegir entre opciones individuales o multijugador.

Explicación del flujo:

1. El jugador abre el menú de selección de modo de juego.
2. Se muestran las opciones disponibles (Modo Progresivo, Modo Estático, Multijugador).
3. El jugador selecciona un modo:
   a. Multijugador: Se presentan opciones para crear o unirse a una sala.
   b. Individual: Se configuran los parámetros del juego.
4. El sistema carga la configuración elegida.
5. La partida comienza según el modo seleccionado

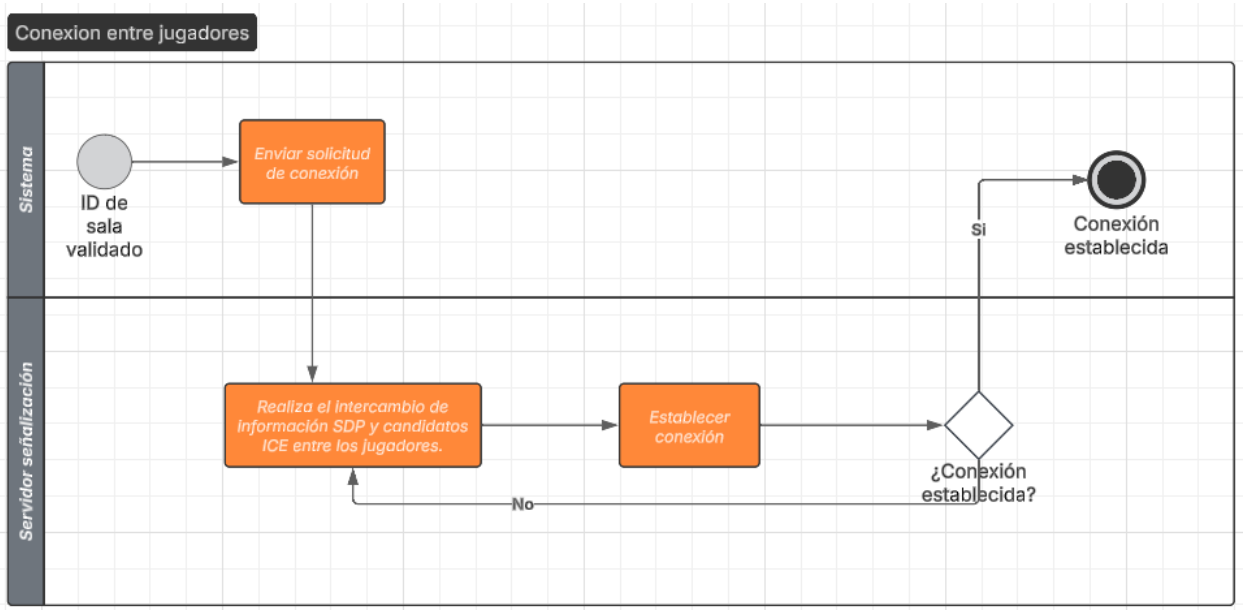

Ilustración 10 Conexión entre jugadores

Descripción General Conexión Entre Jugadores:

Este flujo muestra el proceso de conexión entre dos jugadores en MusicNet, utilizando un servidor de señalización. Una vez que el sistema ha validado el ID de la sala, se inicia el proceso de conexión, donde el servidor de señalización facilita el intercambio de información necesaria para que los jugadores puedan establecer una comunicación directa mediante WebRTC.

Explicación del flujo:

1. El proceso comienza cuando el sistema verifica que el ID de la sala es correcto.
2. El sistema envía una solicitud para iniciar la conexión entre los jugadores.
3. El servidor de señalización facilita el intercambio de información de sesión (SDP) y posibles rutas de conexión (candidatos ICE) entre los jugadores.
4. Se intenta establecer la conexión directa entre los jugadores.
5. Verificación de conexión:
   c. Si la conexión es exitosa → Se marca como Conexión establecida.
   d. Si falla → El proceso puede intentar otro método o reintentar el intercambio.

Diseño Detallado

Esta sección presenta la estructura interna de MusicNet y el comportamiento detallado de sus componentes. Se describe cómo están organizadas las clases y módulos, así como las interacciones clave entre ellos.

a. Estructura del Sistema

MusicNet está compuesto por módulos distribuidos entre el cliente web, el servidor de señalización y el servidor de detección/reconstrucción. A continuación, se describen sus principales componentes y clases.

1. **MusicNet**

   a. **pitchmodule.js**

   - Responsabilidad: Captura audio del micrófono, lo fragmenta en ventanas, lo envía por WebRTC y recibe la nota detectada.
   - Clases/Funciones relevantes:
     - PitchDetector: clase principal encargada de iniciar el procesamiento, configurar WebRTC y gestionar la lógica de envío/recepción.
     - start(), _initWebRTC(), _setupDataChannel(), _sendCurrentWindow()

   b. **scalemapping.js**

   - Responsabilidad: Define y mapea escalas musicales a niveles del juego.
   - Funciones relevantes:
     - generateScale(mode): genera la escala correspondiente al modo seleccionado.
     - playLevel(level): ejecuta la nota esperada.
     - handleDetectedNote(level, noteName, detectionTime): calcula precisión temporal.

   c. **graphics.js**

   - Responsabilidad: Renderiza la escena del juego con Phaser (plataformas, avatares, puntuación).
   - Funciones destacadas: createLevel(), updateJump(), renderScore()

   d. **webrtcmanager.js**

   - Responsabilidad: Maneja la conexión WebRTC entre jugadores.
   - Clases/Funciones relevantes:
     - WebRTCManager.connect(): inicia conexión peer-to-peer.
     - createOfferWhenReady(), sendMessage(), onMessage()

   e. **Servidor de Reconstrucción y Detección (Python)**

   - Archivo principal: service_webrtc.py
   - Responsabilidad: Maneja la recepción de audio en tiempo real desde el navegador usando WebRTC, detecta pérdida de paquetes y aplica reconstrucción con el modelo PARCnet. Luego extrae la frecuencia fundamental de la señal (nota) y la envía de vuelta al cliente por un canal RTCDataChannel.
   - Funciones clave:
     - @app.post("/offer"): establece la sesión WebRTC y canales de comunicación.

- on_track(): gestiona la recepción de audio del cliente.
- stats_loop(): monitorea la tasa de pérdida de paquetes.
- librosa.yin: función utilizada para detectar la nota a partir del audio (pitch detection).

f. Módulo PARCnet (Python)

- Archivo principal: parcnet.py
- Clase principal: PARCnet
  - __init__(): Configura modelos internos (AR y NN) y carga el checkpoint.
  - __call__(): Recibe una señal de audio y una traza de pérdidas, aplica reconstrucción y devuelve la señal completa.
- Dependencias internas:
  - ARModel: predicción lineal autoregresiva.
  - HybridModel: red neuronal convolucional para predicción de señales faltantes.

g. Servidor de Señalización (Node.js)

- Tecnología: Node.js con Socket.IO
- Responsabilidades:
  - Crear, unir y cerrar salas.
  - Gestionar eventos: offer, answer, iceCandidate, createRoom, joinRoom.
  - Promoción automática del host si este se desconecta.

h. Clases complementarias en el cliente

- Tuner: inicializa el micrófono.
- Notes: gestiona la interfaz de selección de notas, validación y visualización.

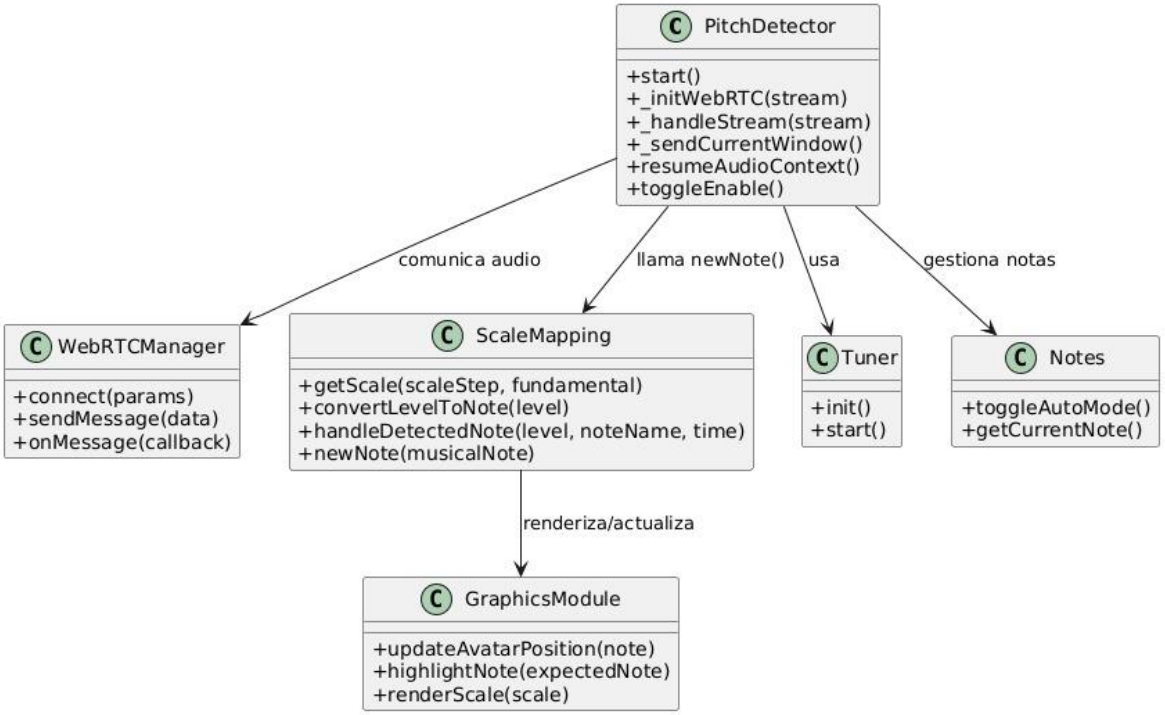

Ilustración 11 Diagrama de clases MusicNet

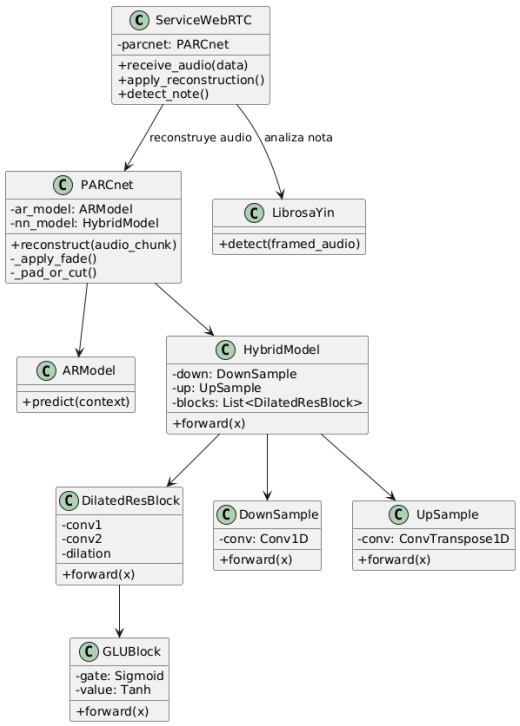

Ilustración 12 Diagrama de clases PARCnet

b. Comportamiento del Sistema

Esta sección describe los procesos complejos del sistema a través de diagramas de secuencia. Se seleccionan las acciones que implican múltiples módulos y comunicación entre componentes.

1. Secuencia: Detección de Nota con Reconstrucción de Paquetes

Descripción: El jugador ejecuta una nota, el audio se transmite al servidor, donde se detecta pérdida, se reconstruye con PARCnet, y se responde con la nota detectada.

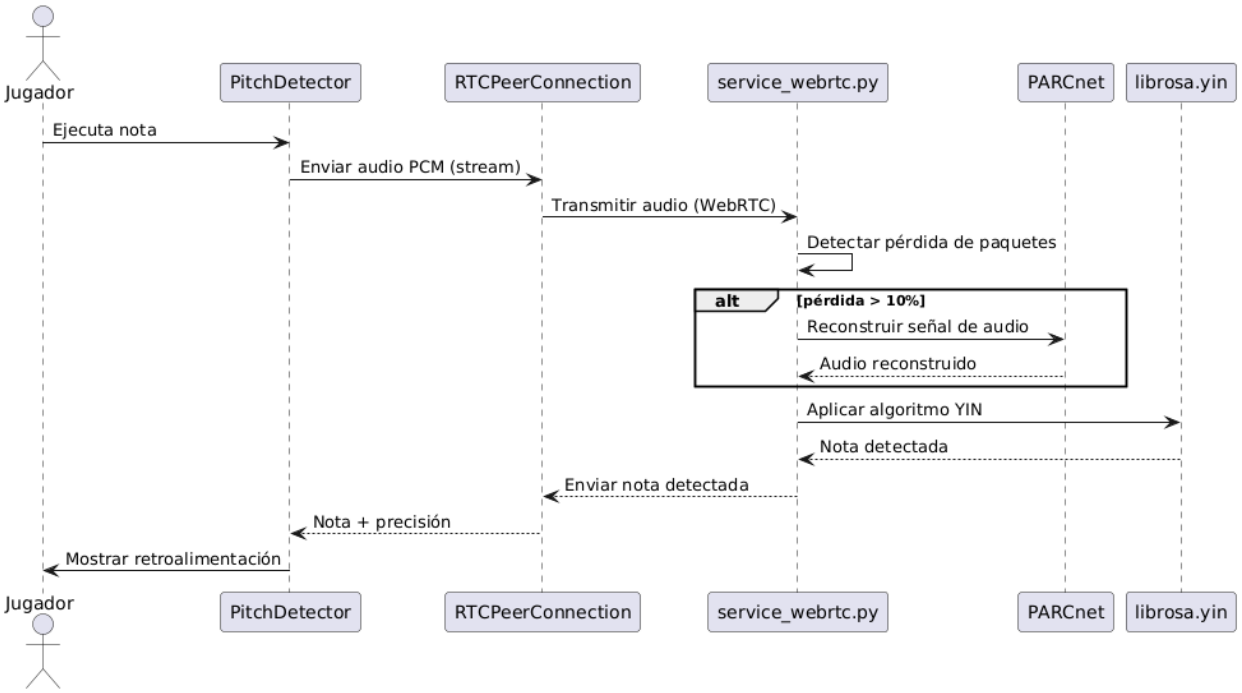

Ilustración 13 Secuencia 1

2. Secuencia: Creación y Sincronización de Sala Multijugador

Descripción: Un jugador crea una sala, otro se une, se establece la conexión WebRTC y se sincronizan los eventos de juego.

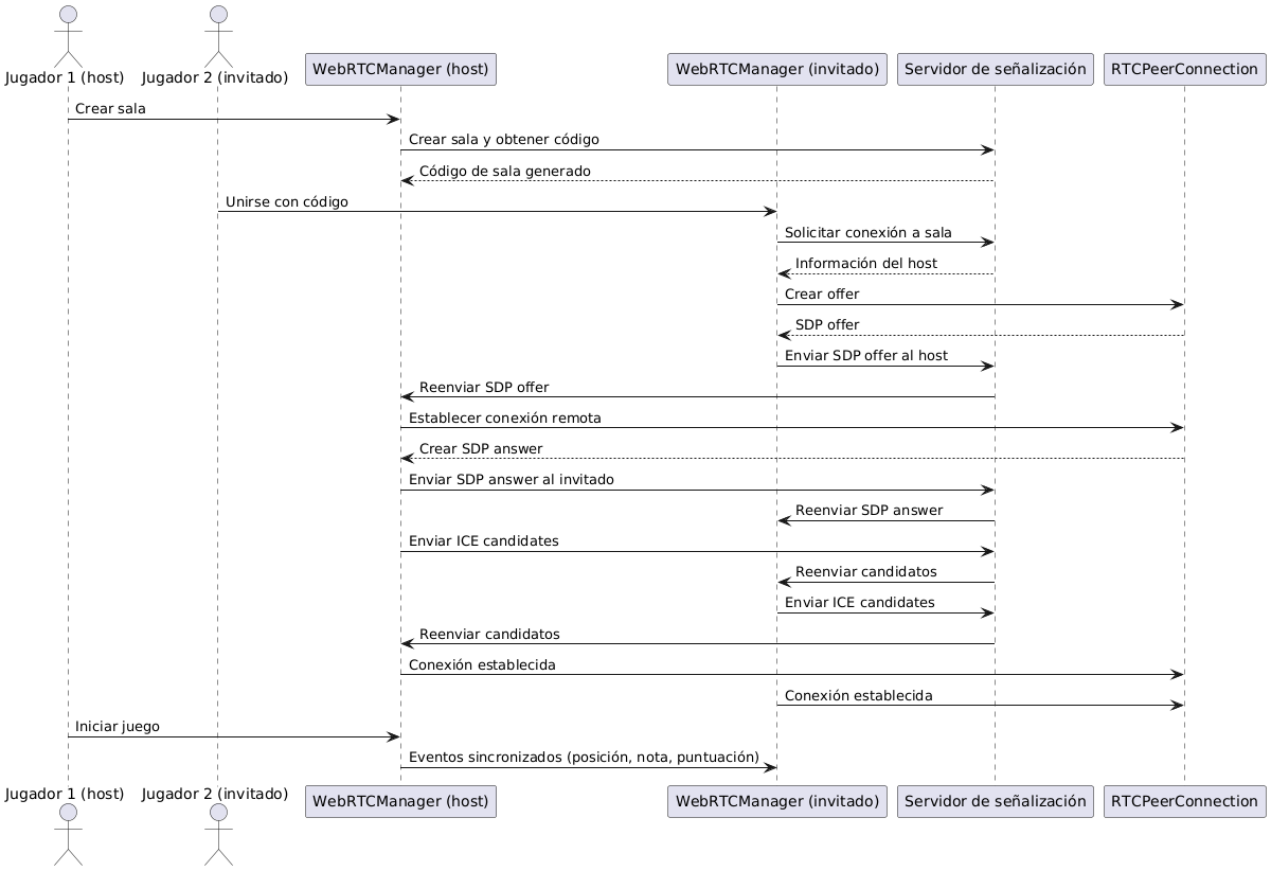

Ilustración 14 Secuencia 2

3. Estado del sistema

Estados principales:

- Esperando selección de escala
- Mostrando nota esperada
- Esperando nota ejecutada
- Evaluando precisión
- Mostrando retroalimentación
- Finalizando sesión

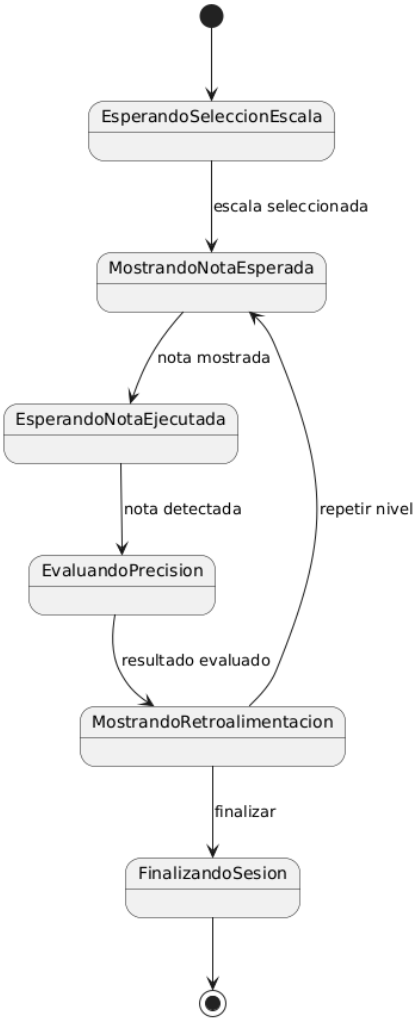

Ilustración 14. Diagrama de estados

c. Interfaz de Usuario

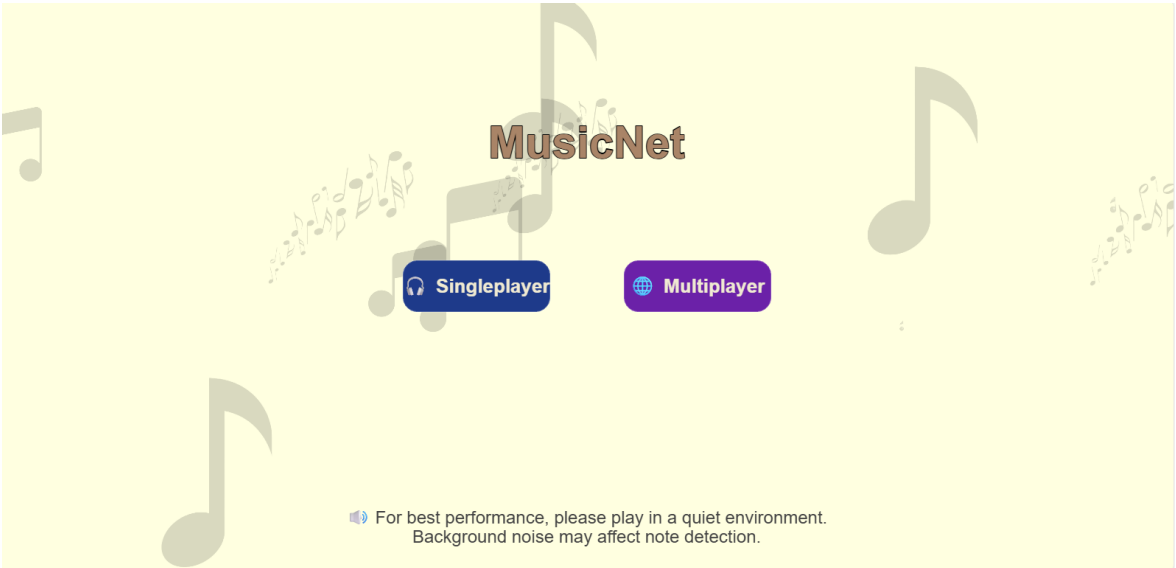

Ilustración 15 Pantalla Selección Modalidad

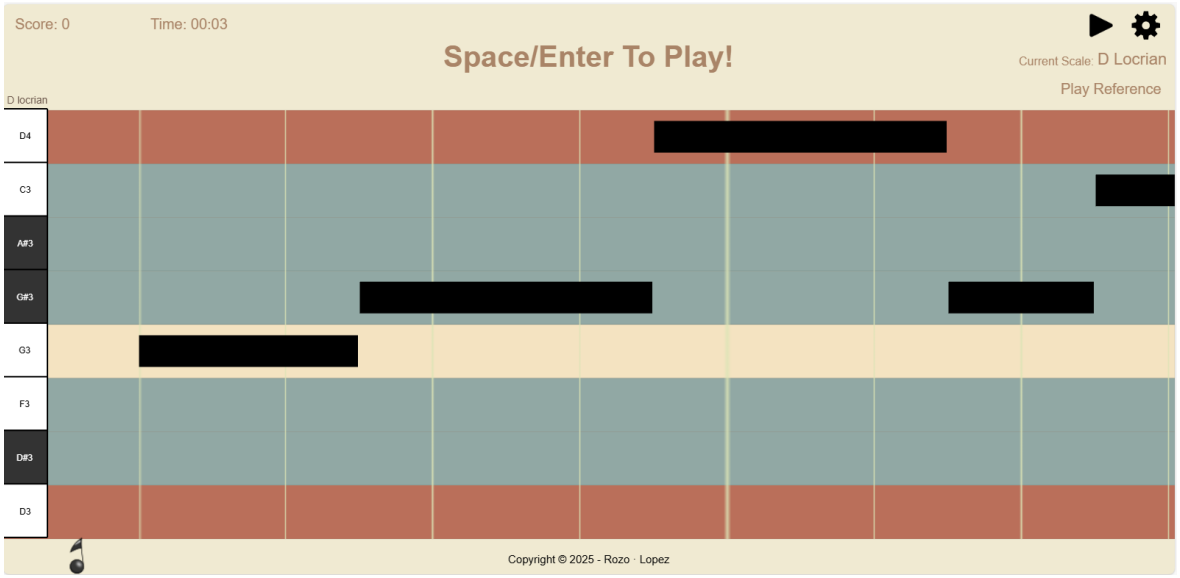

Ilustración 16 Pantalla Juego Individual

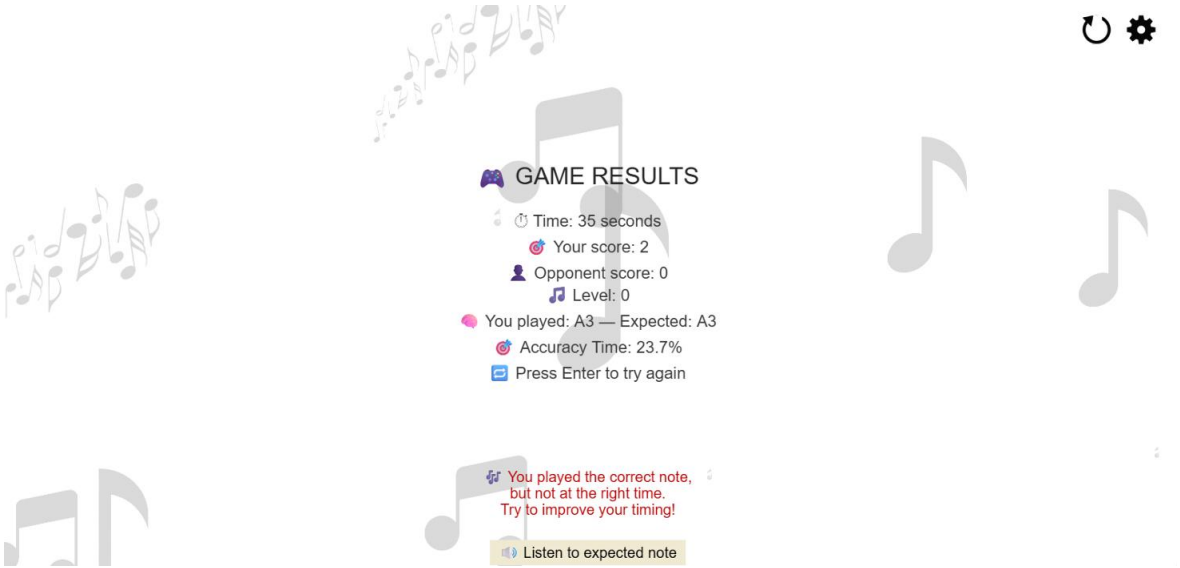

Ilustración 17 Pantalla Resultados Un jugador

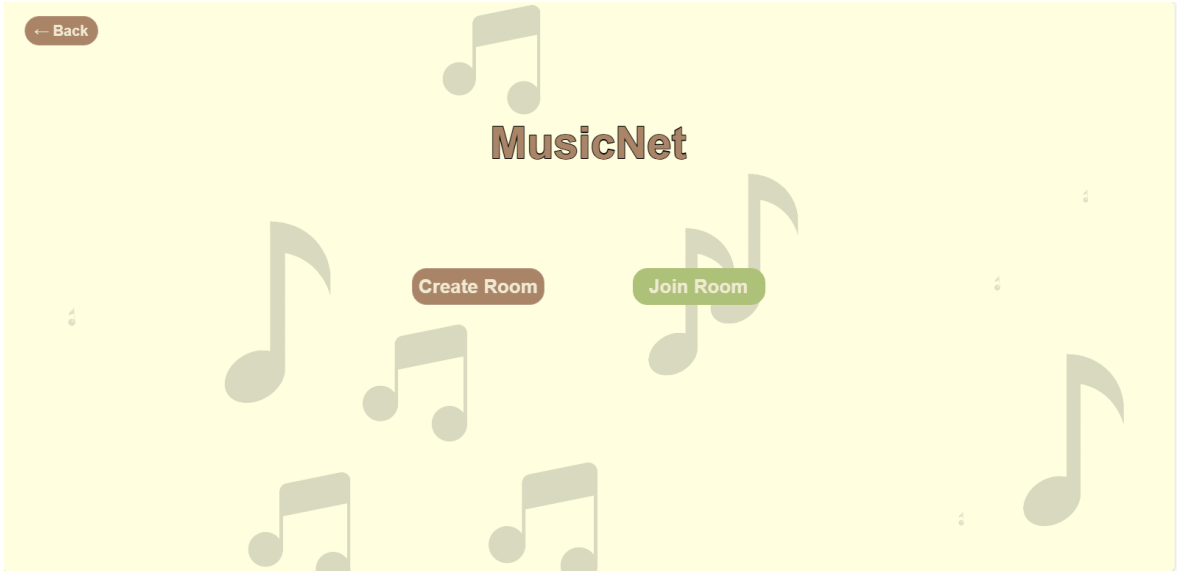

Ilustración 18 Salas Multijugador

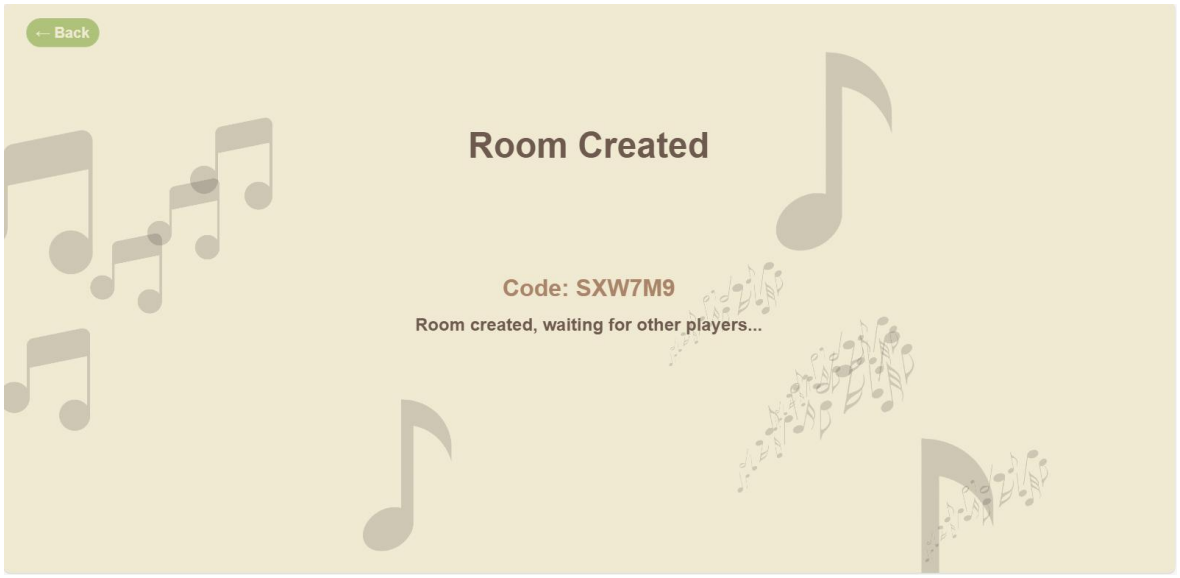

Ilustración 19 Crear Sala (Host)

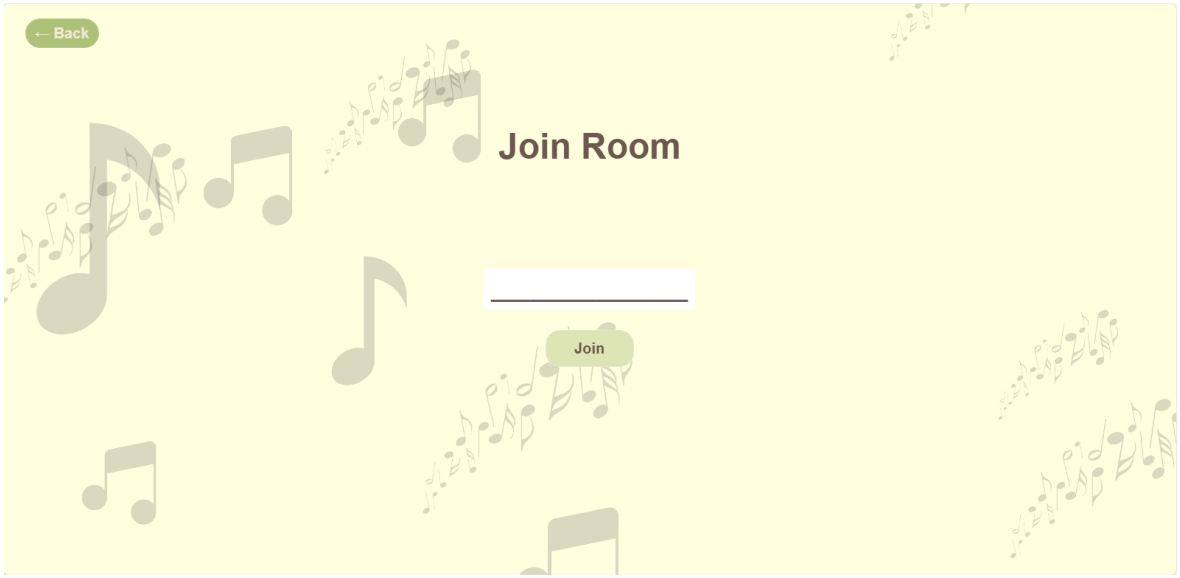

Ilustración 20 Unirse a Sala (Guest)

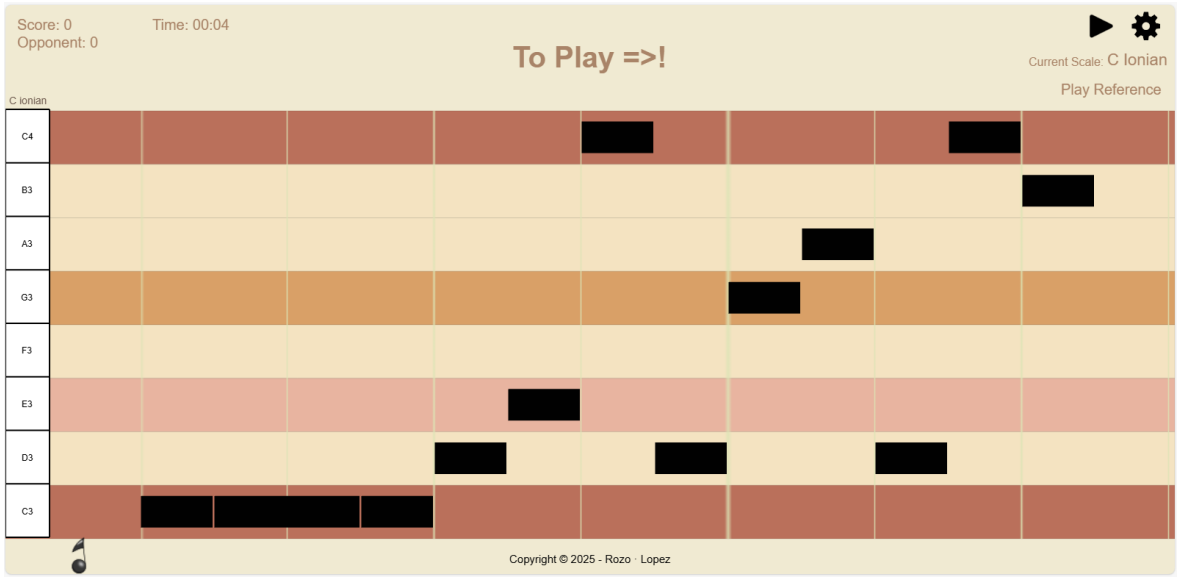

Ilustración 21 Juego Multijugador

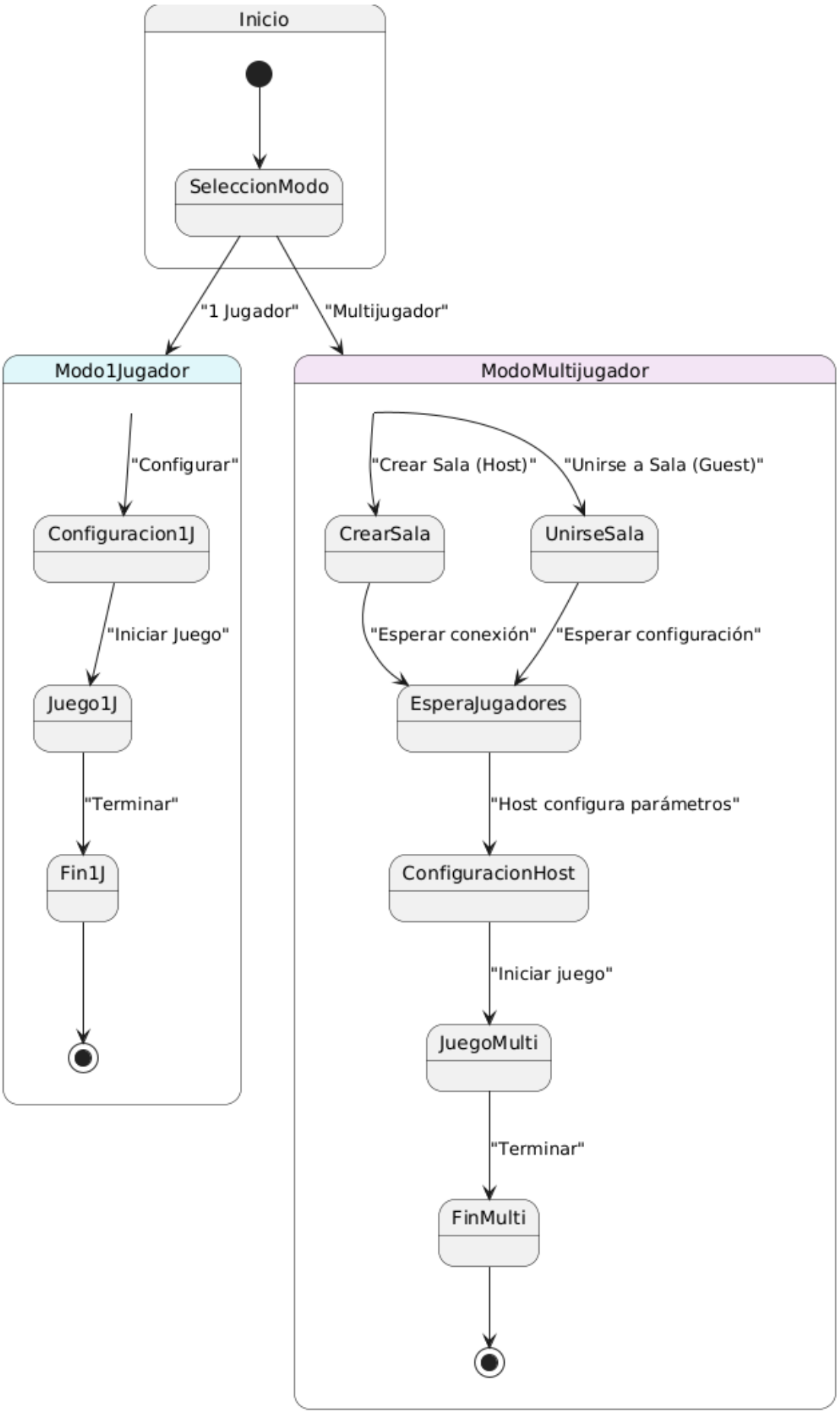

Ilustración 15 Diagrama de Navegación

Referencias

[1] ISO/IEC/IEEE 42010

[2] IEEE Std 1016-2009

[3] RUP. Software Architecture Document http://www.ts.mah.se/RUP/RationalUnifiedProcess/webtmpl/templates/a_and_d/rup_sad.htm

[4] Philippe Kruchten, Architectural Blueprints—The "4+1" View Model of Software Architecture. http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

[5] Scott Ambler, UML Data Modeling Profile. http://www.agiledata.org/essays/umlDataModelingProfile.html

[6] Scott Ambler, User Interface Flow Diagrms. http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm