Plan de Administración de Proyecto

MUSICNET

Plan de Administración de Proyecto

10/05/2025

V.3

Valentina Rozo Gonzalez

Paula Valentina López Cubillos

Historial de Cambios

FECHA DE CAMBIO	VERSIÓN	CAMBIOS	RESPONSABLE
28/10/2024	V.1	Planteamiento original del documento	Valentina Rozo
11/11/2024	V.2	Correcciones de formato e información más clara basados en comentarios de directores	Valentina Rozo
10/05/2025	V.3	Correcciones entrega final	Valentina Rozo

Prefacio

Este documento detalla el proyecto MusicNet, una plataforma interactiva de aprendizaje musical en línea diseñada para mejorar la enseñanza del piano mediante la integración de tecnologías avanzadas de detección de sonido y sincronización en tiempo real. Su propósito es proporcionar una solución educativa innovadora que aborde los desafíos de latencia y pérdida de paquetes en entornos de red inestables, facilitando una experiencia de aprendizaje colaborativo y eficaz para estudiantes de música.

Alcance del Proyecto

El proyecto abarca el diseño, desarrollo, implementación y pruebas de MusicNet, incluyendo funcionalidades como la detección precisa de notas musicales, un modo multijugador, y un sistema de compensación de pérdida de datos para asegurar la calidad del audio en tiempo real.

Audiencia del Documento

Este documento está dirigido a los directores del proyecto, desarrolladores, diseñadores y cualquier otro profesional involucrado en el desarrollo y evaluación de MusicNet. Es especialmente útil para los futuros integrantes del equipo, quienes podrán comprender la estructura y los objetivos del proyecto, así como los requisitos técnicos y de implementación establecidos.

Contenido

Lista de Ilustraciones

Lista de Tablas

Vista General del Proyecto

6.1 Visión del Producto

MusicNet es una plataforma educativa diseñada para transformar la enseñanza del piano en línea. A través de un entorno interactivo, los estudiantes podrán practicar en tiempo real, ya sea en solitario o en un entorno multijugador competitivo. MusicNet solucionará los problemas técnicos que afectan las aplicaciones musicales en tiempo real, como la latencia, la pérdida de paquetes y la sincronización. Utilizando inteligencia artificial (IA) para la compensación de pérdida de paquetes y herramientas avanzadas de detección de sonido, la plataforma proporcionará una experiencia educativa inmersiva y sin interrupciones. Además, se integrarán elementos de gamificación que fomenten la motivación de los estudiantes, brindando retroalimentación instantánea y una interfaz intuitiva. Con MusicNet, tanto principiantes como músicos experimentados podrán mejorar sus habilidades musicales sin las barreras tradicionales de la enseñanza a distancia.

6.2 Propósito, Alcance y Objetivos

Propósito: El propósito de **MusicNet** es proporcionar una solución innovadora a los desafíos de la enseñanza musical en línea, especialmente en la enseñanza del piano. Los sistemas actuales a menudo fallan en proporcionar una experiencia fluida debido a problemas de latencia y pérdida de paquetes, lo que impacta negativamente en la precisión musical y la sincronización entre el estudiante y el software. MusicNet busca abordar estos problemas mediante la implementación de IA para la compensación de pérdida de paquetes, junto con un módulo avanzado de detección de sonidos, asegurando una experiencia de enseñanza confiable y de alta calidad.

El propósito de este proyecto es también promover la democratización del acceso a la práctica musical, permitiendo que estudiantes de diferentes contextos puedan mejorar sus habilidades musicales sin las barreras físicas de la enseñanza tradicional.

Alcance: El alcance de MusicNet incluye el desarrollo de una plataforma web accesible desde cualquier dispositivo con conexión a internet y un piano digital o MIDI. Las principales funcionalidades de la plataforma son:

1. **Modalidad multijugador**: Los estudiantes podrán competir entre sí en tiempo real, con visualización simultánea del progreso de sus compañeros, lo que fomenta un aprendizaje colaborativo y competitivo.
2. **Interfaz gráfica interactiva**: La plataforma contará con una interfaz visualmente atractiva y fácil de usar, diseñada para guiar a los estudiantes a lo largo de sus lecciones y brindar retroalimentación instantánea sobre sus ejecuciones.
3. **Detección de sonidos en tiempo real**: Utilizando **AubioJS**, MusicNet permitirá la detección precisa de las notas musicales que los estudiantes

toquen en su piano, lo que facilitará el aprendizaje a través de ejercicios auditivos y visuales.
4. **Compensación de pérdida de paquetes**: MusicNet integrará IA para reducir los efectos de la pérdida de paquetes en redes inestables, lo que asegurará que la transmisión de audio sea constante y sin interrupciones.
5. **Retroalimentación**: Se proporcionará retroalimentación continua que permitirá a los participantes conocer su progreso y recibir orientación sobre cómo mejorar, haciendo más dinámica y atractiva la experiencia de aprendizaje al permitirles visualizar su mejora en tiempo real.

Objetivos:

1. **Implementar la modalidad multijugador**: Desarrollar una función que permita a dos jugadores competir simultáneamente utilizando su propio piano, con visualización en tiempo real del progreso de cada jugador. Esta modalidad fomentará tanto la colaboración como la competencia amistosa, ofreciendo una nueva manera de practicar y aprender piano de manera interactiva.
2. **Desarrollar interfaces gráficas interactivas**: Crear una interfaz visualmente atractiva e intuitiva que facilite la interacción de los usuarios con la aplicación. La interfaz incluirá pistas de notas dinámicas, avatares musicales, y retroalimentación visual y auditiva que permitirá a los estudiantes comprender y corregir sus errores.
3. **Implementar un sistema de compensación de pérdida de paquetes**: Analizar y seleccionar los modelos de IA propuestos en el **IEEEIS^2 2024 Music Packet Loss Concealment Challenge** para integrarlos en la plataforma. Esto permitirá una reducción significativa de la latencia en redes inestables, asegurando la continuidad del aprendizaje musical en tiempo real.
4. **Desarrollar un módulo de detección y sincronización de sonidos**: Utilizar tecnologías como **AubioJS** para la detección precisa de notas y ritmos ejecutados por los estudiantes en tiempo real. El módulo deberá procesar la información auditiva con precisión y sincronizarla con las plataformas gráficas en la pantalla.
5. **Realizar pruebas exhaustivas de la plataforma**: Evaluar el rendimiento de MusicNet en condiciones de red variables, probando la estabilidad del sistema multijugador y la precisión de la detección de sonido bajo distintas circunstancias. Se implementarán ajustes basados en los resultados de las pruebas para asegurar que la experiencia del usuario sea óptima.
6. **Promover el uso de MusicNet en instituciones educativas**: A largo plazo, uno de los objetivos es expandir el uso de la plataforma en escuelas y academias de música, ofreciendo un nuevo enfoque a la enseñanza del piano que aprovecha las tecnologías más avanzadas en el procesamiento de audio y la inteligencia artificial.

6.3 Supuestos y Restricciones

Supuestos:

- **Disponibilidad de pianos digitales:** Se asume que los usuarios de la plataforma tendrán acceso a pianos digitales compatibles que puedan conectarse a MusicNet para ejecutar las lecciones.
- **Acceso a conexión a internet estable:** Si bien MusicNet está diseñada para mitigar problemas relacionados con la pérdida de paquetes, se asume que los usuarios contarán con acceso a una conexión a internet razonablemente estable para una experiencia fluida.
- **Acceso a dispositivos con navegador web:** MusicNet estará disponible para su uso en cualquier dispositivo con un navegador web moderno. Se asume que los usuarios contarán con computadoras o tabletas compatibles que soporten la ejecución de la plataforma.
- **Capacitación de los usuarios en herramientas básicas de informática:** Se asume que los usuarios tendrán conocimientos básicos de informática y navegación por internet, facilitando la adopción de la plataforma sin requerir capacitaciones adicionales.
- **Formación Musical de los Usuarios:** Se asume que los usuarios tienen un conocimiento básico de teoría musical, suficiente para comprender escalas, tonos y ritmos, lo cual es fundamental para utilizar MusicNet de manera efectiva.

Restricciones:

Dependencia de la calidad de la conexión a internet: Aunque se utilizarán modelos predictivos de IA para compensar la pérdida de paquetes y mejorar la experiencia de usuario en redes inestables, MusicNet podría verse limitada en redes extremadamente deficientes o en situaciones de desconexión prolongada. Se establece un límite mínimo en la velocidad de conexión a internet requerida para la plataforma. Para garantizar la calidad del audio y la sincronización, se recomienda una conexión de al menos 5 Mbps de velocidad de subida, una velocidad de descarga de 15-25 Mbps.

Requerimientos técnicos: El uso de AubioJS para la detección de notas y los modelos de IA para la compensación de pérdida de paquetes requerirán ciertos recursos del dispositivo que los usuarios utilicen.

- **Técnicas de Ocultamiento de Pérdida de Paquetes (PLC)**

  En un análisis preliminar de los resultados del IEEE-IS^2 2024 Music Packet Loss Concealment Challenge, se evidencio que hay un sistema se destaca por su rendimiento en métricas objetivas y pruebas de escucha subjetiva, siendo efectivo en condiciones de pérdida de paquetes, manejando interrupciones de hasta 50 paquetes consecutivos.

  - Tasa de Muestreo: 44.1 kHz, procesando paquetes de 512 muestras (11.6 ms). Esto permite una alta calidad en la transmisión de audio en tiempo real.
  - Latencia Baja: PARCnet-IS^2 es un sistema causal, lo que significa que no utiliza datos futuros para la predicción de paquetes,

característica clave para aplicaciones de música en red.
- Pérdida de Paquetes Controlada: Aunque el sistema soporta interrupciones, es más preciso con pérdidas cortas (hasta 6 paquetes), y también es funcional con pérdidas de hasta 16 paquetes.

- **Rangos de Latencia para MusicNet**

  Para la aplicación de enseñanza musical en red, se establecen los siguientes niveles de latencia para una sincronización óptima:

  - Latencia Ideal: Menor a 20 ms. Este nivel es casi imperceptible, proporcionando una sincronización en tiempo real fluida y esencial para el entrenamiento musical interactivo.
  - Latencia Aceptable: Entre 20 y 50 ms. Aunque perceptible en algunos casos, aún permite una buena interacción, siendo adecuado para la mayoría de las funciones de MusicNet.
  - Latencia Máxima Tolerable: Hasta 100 ms. Superar este rango puede dificultar la precisión en la sincronización musical, especialmente en ejercicios de ritmo y modos de competencia.

  Estos valores servirán como guía para asegurar la mejor experiencia de usuario en la enseñanza musical de MusicNet, optimizando la latencia para la interacción en tiempo real.

Tiempo limitado para pruebas: Dado el tiempo limitado para el desarrollo, las pruebas de rendimiento y optimización del sistema se realizarán en un conjunto reducido de dispositivos y entornos de red, lo que podría limitar la capacidad de prever todos los escenarios de uso posibles.

6.4 Entregables

El proyecto MusicNet contempla una serie de entregables que se desarrollan y entregan en diferentes etapas del proceso, dirigidos a distintos actores clave del proyecto: directores, jurados, estudiantes y docentes. Estos entregables abarcan desde documentos técnicos hasta productos funcionales y materiales de apoyo.

Nombre del entregable	**Descripción**	**Dirigido a**	**Fechas de entrega**
Propuesta para Proyecto de Grado	Documento inicial aprobado, con planteamiento del problema, objetivos, metodología	Directores, Jurado	26/05/2025
Especificación de Requisitos de Software	Documento con los requisitos funcionales, no funcionales y restricciones del sistema.	Directores, Jurado	11/11/2024 21/02/2025 19/05/2025 26/05/2025
Plan de Administración del	Documento que describe el cronograma, gestión de recursos,	Directores, Jurado.	11/11/2024 19/05/2025

Proyecto	riesgos y organización del equipo.		26/05/2025
Descripción del Diseño del Software	Documento que incluye la arquitectura del sistema, diagramas UML, diseño de interfaz y lógica de componentes.	Directores, Jurado.	11/11/2024 21/02/2025 19/05/2025 26/05/2025
Manual del usuario	Guía de uso práctica para estudiantes y docentes; explicada también en el video demostrativo.	Estudiantes, Docentes	05/05/2025 19/05/2025 26/05/2025
Sistema funcional: Plataforma MusicNet	Sistema multijugador funcional con detección de sonidos, sincronización y compensación de pérdida de paquetes.	Estudiantes, Docentes	05/05/2025
Código fuente documentado de MusicNet	Código organizado, comentado y con documentación de instalación, estructura y dependencias.	Directores, Jurado	19/05/2025 26/05/2025
Documento de calidad de pruebas	E	Directores, Jurado	19/05/2025 26/05/2025
Memoria del proyecto	Documento final que resume el proceso completo, resultados técnicos, análisis y conclusiones.	Directores, Jurado	19/05/2025 26/05/2025
Diapositivas de sustentación del proyecto	Presentación visual para la exposición oral ante los jurados, resumiendo el proceso, resultados y conclusiones del proyecto.	Directores, Jurado	19/05/2025 26/05/2025

Tabla 1. Entregables

Descripción de los entregables principales:

- **Propuesta para Proyecto de Grado:** Documento inicial aprobado que establece la justificación del proyecto, los objetivos, el planteamiento del problema, el marco metodológico y el cronograma tentativo de desarrollo.
- **Especificación de Requisitos de Software:** Documento que define detalladamente los requisitos funcionales y no funcionales del sistema, incluyendo restricciones técnicas, prioridades de desarrollo y criterios de aceptación.
- **Plan de Administración del Proyecto:** Documento que describe la planificación integral del proyecto, incluyendo la asignación de recursos, cronograma detallado, análisis de riesgos y estrategias de mitigación.
- **Descripción del Diseño del Software:** Contiene la arquitectura lógica del sistema, diagramas UML (casos de uso, componentes, clases, despliegue), el diseño de la interfaz gráfica y las decisiones técnicas adoptadas en el desarrollo.
- **Sistema funcional (Plataforma MusicNet v1.0):** Versión operativa inicial del sistema implementado, con funcionalidades clave como multijugador en tiempo real, detección de notas musicales, sincronización entre usuarios y

tolerancia a pérdida de paquetes en redes inestables.

- **Código Fuente Documentado:** Repositorio completo del código del sistema, con estructura organizada, comentarios explicativos, scripts de instalación y archivos de configuración necesarios para el despliegue.
- **Documento de Calidad de Pruebas:** Informe que describe la planificación, ejecución y resultados de las pruebas realizadas sobre el sistema. Incluye pruebas de funcionalidad, rendimiento, estrés, carga, y estabilidad en distintas condiciones de red.
- **Memoria del Proyecto:** Documento final que integra todos los aspectos del desarrollo del sistema: contexto, metodología, arquitectura, resultados de pruebas, conclusiones y recomendaciones. Sirve como soporte principal para la evaluación del proyecto.
- **Manual del Usuario:** Guía práctica destinada a estudiantes y docentes para facilitar el uso de la plataforma. Incluye instrucciones sobre el acceso, la navegación y el uso de las funcionalidades clave. Este manual se complementa con un video interactivo que muestra el funcionamiento completo del sistema desde la perspectiva del usuario final. Sirve como evidencia funcional y también como soporte visual para la sustentación del proyecto.
- **Diapositivas de Sustentación:** Presentación diseñada para exponer el proyecto ante los jurados, que resume los principales aspectos técnicos, logros, resultados y conclusiones obtenidas durante el desarrollo.

6.5 Resumen de Calendarización y Presupuesto

El proyecto **MusicNet** seguirá una estructura de fases que abarca desde el análisis de requisitos hasta las pruebas finales y la documentación. A continuación, se presenta un resumen detallado de las fases principales del proyecto, junto con la distribución del presupuesto estimado para cada una de ellas.

Fase	Descripción	Duración	Fecha de Inicio	Fecha de Finalización
Fase 1: Análisis de Requisitos	Identificación y documentación de funcionalidades clave; reunión inicial con directores para revisar requisitos.	2 semanas	27 de enero	9 de febrero
Fase 2: Diseño de Arquitectura e Interfaz	Diseño de la arquitectura técnica y creación de prototipos de interfaz; revisión con directores y usuarios clave.	2 semanas	10 de febrero	23 de febrero
Fase 3: Desarrollo del Sistema	Desarrollo de funcionalidades principales; implementación del sistema multijugador y	9 semanas	24 de febrero	27 de abril

	detección de sonido.			
Fase 4: Pruebas e Integración	Pruebas unitarias y de rendimiento; optimización del sistema.	2 semanas	28 de abril	11 de mayo
Fase 5: Documentación y Ajustes Finales	Creación de documentos finales y ajustes en el sistema según los resultados de las pruebas.	1 semana	12 de mayo	19 de mayo

Tabla 2.Fases e Hitos del Proyecto

6.6 Evolución del Plan

El plan de proyecto de ***MusicNet*** evolucionara progresivamente mediante una metodología ágil.[11] La planificación y ejecución se organizaron en torno a sprints, con una combinación estructurada de herramientas, reuniones y revisiones colaborativas.

Estructura de trabajo semanal

Durante el desarrollo del proyecto, se mantendrá un ciclo de trabajo constante con las siguientes actividades clave:

- **Reuniones internas semanales**: Todos los lunes, el equipo de trabajo realizara una reunión de coordinación para revisar el estado general de las tareas registradas en Trello. En estas sesiones se evaluarán los avances, se reorganizaban prioridades y se identificaran subtareas derivadas de los objetivos de cada sprint.
- **Gestión en Jira y estructura de sprints**: La parte de desarrollo técnico se gestionará con Jira, donde se estructuran sprints semanales, según las necesidades del proyecto y la carga de trabajo:
  Cada sprint se organizará a partir de scrums, basados en los objetivos definidos en la Propuesta de Proyecto. Si se identifican tareas más específicas, se crearán subtareas asociadas a cada scrum para facilitar el seguimiento granular.
- **Asignación dinámica de tareas**: Las asignaciones se ajustarán en tiempo real de acuerdo con el feedback de los directores, recibido en reuniones anteriores, y conforme a las asignaciones pendientes identificadas en Jira. Esto permitirá una trazabilidad clara entre lo planeado y lo ejecutado.
- **Reuniones de seguimiento con directores**: Todos los viernes se llevarán a cabo reuniones de presentación con los directores del proyecto. En estas sesiones se socializarán los **avances semanales**, se mostrarán prototipos o funcionalidades implementadas, y se recibirá retroalimentación que influirá directamente en los ajustes del siguiente sprint.

Adaptación a resultados de pruebas

El plan también se ajustará en función de los resultados obtenidos durante las pruebas. Si se detectan fallos críticos en etapas de integración, latencia o estabilidad, se redistribuirá el esfuerzo del equipo para priorizar la resolución de estos problemas sin comprometer los entregables ni la calidad final del producto.

Este enfoque iterativo, guiado por evidencia y feedback continuo, permitirá una evolución orgánica del proyecto, asegurando alineación constante entre los

objetivos planteados, el cronograma del PMP y las expectativas de los directores y usuarios finales.

Glosario

El glosario del proyecto **MusicNet** incluye los términos clave y técnicos que serán utilizados a lo largo del proyecto. Esto asegura que todos los miembros del equipo, los directores y los usuarios finales tengan una comprensión común de los términos relacionados con la plataforma.

1. AubioJS: Una biblioteca de JavaScript utilizada para la detección de audio en tiempo real, basada en la biblioteca de procesamiento de audio Aubio (originalmente en Python). AubioJS permite detectar tonos y frecuencias, convirtiéndolas en notas musicales para su procesamiento en navegadores web. [1].
2. Latencia: En aplicaciones de transmisión en tiempo real, la latencia es el retraso que ocurre entre el momento en que se produce un sonido y el momento en que se escucha o se procesa. En MusicNet, una latencia baja es crucial para mantener la sincronización musical. [2].
3. Pérdida de Paquetes: Fenómeno en redes donde algunos paquetes de datos no llegan a su destino, causando interrupciones en la transmisión de audio o video. En MusicNet, se implementa PARCnet-IS^2 para mitigar la pérdida de paquetes y preservar la calidad de la transmisión de sonido en tiempo real. [3].
4. Tasa de Muestreo: Número de muestras de audio que se capturan por segundo. En el caso de MusicNet, PARCnet-IS^2 opera a una tasa de 44.1 kHz, una frecuencia común en grabaciones de alta calidad que asegura la precisión en la transmisión de música. [4].
5. Interfaz de Usuario (UI)
   Conjunto de elementos visuales e interactivos con los que un usuario interactúa en la plataforma MusicNet. La interfaz está diseñada para ser intuitiva, permitiendo a los usuarios ejecutar notas musicales y recibir retroalimentación de su rendimiento. [5].
6. Experiencia de Usuario (UX): Se refiere a la percepción y satisfacción del usuario al interactuar con una aplicación. La UX en MusicNet está optimizada para asegurar que los estudiantes de música puedan acceder y navegar fácilmente por la plataforma mientras practican y aprenden. [5].
7. Control de Versiones: Sistema para gestionar y registrar cambios en el código fuente, permitiendo colaborar y rastrear modificaciones. En MusicNet, se utiliza GitHub como repositorio de control de versiones para almacenar el código y documentar el desarrollo del proyecto. [6].
8. Metodología Ágil: Conjunto de principios para la gestión y desarrollo de proyectos que priorizan la flexibilidad y la colaboración. MusicNet se desarrolla utilizando una combinación de Scrum y Kanban para organizar el trabajo en sprints y visualizar el flujo de tareas. [7][8].
9. Procesamiento de Señales: Técnica utilizada para analizar, modificar o sintetizar señales, como el audio en aplicaciones musicales. MusicNet emplea

técnicas de procesamiento de señales para capturar y sincronizar audio en tiempo real. [9].

10. Git: Herramienta de control de versiones que permite a los desarrolladores gestionar el historial de cambios en el código. MusicNet utiliza Git para asegurar que el trabajo del equipo esté bien documentado y que las modificaciones puedan ser revertidas o revisadas fácilmente. [6].

Contexto del proyecto

8.1 Modelo de Ciclo de Vida

El ciclo de vida del desarrollo del proyecto *MusicNet* seguirá un **enfoque ágil híbrido**, combinando metodologías **Scrum** y **Kanban**. Esta combinación permitirá al equipo adaptarse dinámicamente a los cambios del proyecto, con entregas incrementales de valor y seguimiento continuo mediante herramientas de gestión visual como **Trello** y **Jira**. Se desarrollarán **sprints**, cada uno con objetivos específicos derivados del backlog principal y subdivididos en scrums y subtareas según las necesidades del equipo.

Durante todo el desarrollo, se mantendrá una rutina fija:

- **Reuniones internas cada lunes** para revisar el avance general, reorganizar prioridades, y ajustar las tareas en Trello.
- **Reuniones con los directores cada viernes**, donde se presentarán avances tangibles, se obtendrá feedback y se alinearan entregables con los hitos del proyecto.

La siguiente tabla resume las principales fases del ciclo de vida, sus actividades asociadas y entregables clave:

Fase	Descripción	Hitos y Entregables
Inicio del Proyecto	Se realizará la planificación inicial del proyecto, se creará el backlog en Trello y Jira, y se definirá la arquitectura base del sistema	Propuesta de Proyecto de Grado Backlog de producto Arquitectura del sistema
Análisis de Requerimientos y Diseño	Análisis detallado de requerimientos y diseño técnico. Se desarrollarán prototipos de interfaz y se ajustara el diseño según comentarios de los directores.	Especificación de Requerimientos de Software Prototipos UI/UX Diseño Arquitectónico
Desarrollo Incremental	Implementación progresiva durante sprints, con seguimiento diario del avance. Las tareas se dividirán en Jira por scrums y subtareas.	Funcionalidades clave implementadas Versión preliminar de la plataforma Código documentado
Pruebas e Integración	Pruebas unitarias y rendimiento. Se adaptarán los tiempos de desarrollo a los resultados de pruebas en red.	Pruebas unitarias Pruebas de rendimiento en redes inestables
Despliegue y	Se desplegará la plataforma, se entregará la	Plataforma MusicNet en

Entrega	versión funcional y se elaborará toda la documentación requerida. También se realizarán pruebas reales con usuarios	funcionamiento Manual de Usuario Documentación Técnica Memoria del Proyecto

Tabla 3. Principales fases del ciclo de vida

Diagrama BPMN

A continuación, se incluirán un diagrama BPMN y una figura que ilustre la idea general del ciclo de vida del producto:

- Diagrama BPMN: Este diagrama mostrará el flujo de trabajo del ciclo de vida del desarrollo del proyecto, resaltando las interacciones entre las fases y los roles involucrados.

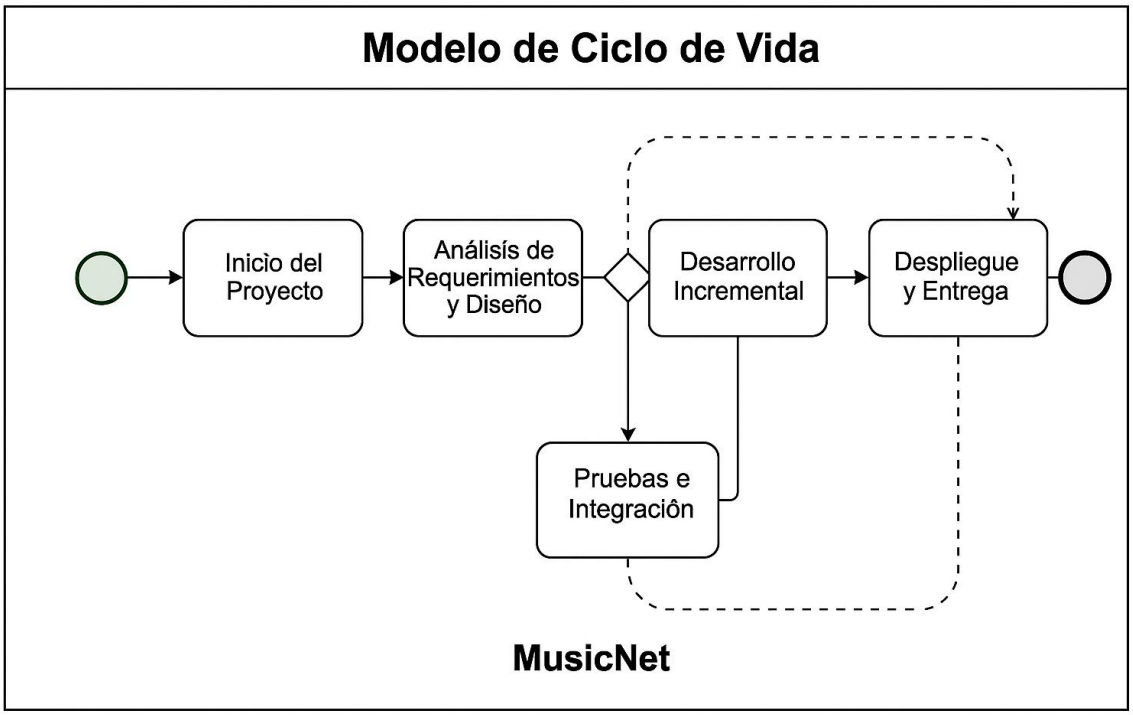

Ilustración 1. BPMN Ciclo de vida

8.1.1 Análisis de Alternativas y Justificación

En el contexto de **MusicNet**, se evaluaron varias alternativas en términos de metodología de desarrollo, tecnologías a utilizar, herramientas de detección de sonido y sistemas de gestión de red para compensación de pérdida de paquetes. A continuación, se presenta un análisis detallado de las alternativas consideradas y la justificación de las elecciones finales.

1. **Alternativas de Metodología de Desarrollo:**

   - **Opción 1: Waterfall (Ciclo de Vida en Cascada)**: La metodología tradicional en cascada sigue un enfoque secuencial, donde cada fase del proyecto debe completarse antes de pasar a la siguiente. Aunque este enfoque puede ser útil en proyectos bien definidos, no ofrece la

flexibilidad necesaria para un proyecto que involucra múltiples cambios e iteraciones, como **MusicNet**. Además, la naturaleza de la enseñanza musical en línea exige una respuesta rápida a las necesidades cambiantes de los usuarios.

- **Opción 2: Agile (Scrum y Kanban)**: El enfoque ágil [14] es más adecuado para proyectos donde los requisitos pueden evolucionar. Scrum proporciona estructura mediante sprints, lo que permite al equipo trabajar en partes del proyecto de manera iterativa y receptiva a los comentarios. Kanban, por su parte, facilita la visualización del flujo de trabajo, permitiendo al equipo gestionar tareas y prioridades en tiempo real. Esta combinación de metodologías se ha adoptado para **MusicNet** debido a su adaptabilidad a los cambios, lo que garantiza que el proyecto se mantenga alineado con las necesidades de los usuarios y el entorno de desarrollo.

Justificación: El enfoque ágil permitirá al equipo realizar entregas incrementales y recibir feedback continuo de los usuarios, ajustando el desarrollo según sea necesario. Esto es crítico para la funcionalidad de **MusicNet**, que se basa en la interacción en tiempo real y la detección de sonidos.

2. **Alternativas Tecnológicas:**

- **Opción 1: Detección de Sonido con Herramientas Nativas**: El uso de herramientas nativas de JavaScript podría haber sido considerado para la detección de sonido, sin embargo, esto podría limitar la precisión y el rendimiento del sistema. Aunque hay bibliotecas disponibles, carecen de las características avanzadas que se requieren para una aplicación de música en tiempo real.
- **Opción 2: AubioJS** [1] es una biblioteca específicamente diseñada para la detección de tonos y ritmos en tiempo real. Su uso facilitará una integración fluida con la plataforma y proporcionará la precisión necesaria para la experiencia de usuario. Esta opción permite a **MusicNet** realizar detecciones de sonido con alta precisión y en tiempo real.

Justificación: La elección de **AubioJS** es fundamental para el éxito del proyecto, ya que proporciona la base necesaria para la detección de notas, permitiendo que los estudiantes reciban retroalimentación instantánea sobre su ejecución musical.

3. **Alternativas de Compensación de Pérdida de Paquetes:**

- **Opción 1 Soluciones Básicas de Compensación:** Las soluciones básicas de compensación de pérdida de paquetes, como el uso de buffers estáticos, no son suficientes para un entorno musical donde la latencia y la sincronización son críticas. Este enfoque podría llevar a desincronización y distorsiones sonoras.

- **Opción 2 Modelo Predictivo PARCNet del IEEEIS² 2024:** Se adoptará el modelo predictivo PARCNet presentado en el **IEEEIS² 2024 Music Packet Loss Concealment Challenge**. Este modelo es capaz de predecir y reconstruir datos perdidos de manera efectiva, mejorando significativamente la experiencia de usuario en condiciones de red inestables. [3]

Justificación: La integración de modelos predictivos de IA para la compensación de pérdida de paquetes representa un avance significativo en la calidad del audio transmitido. Esto asegurará que los estudiantes tengan una experiencia de aprendizaje fluida, sin interrupciones que podrían afectar su progreso.

4. **Herramientas de Desarrollo:**

- **Opción 1 Herramientas de Gestión Tradicionales**: Aunque herramientas como Microsoft Project pueden ser útiles, no ofrecen la flexibilidad ni la capacidad de visualización requeridas para un entorno ágil. La falta de integración con metodologías como Scrum y Kanban limita su eficacia en un proyecto como **MusicNet**.
- **Opción 2 Trello y Jira**: La selección de **Trello** para la gestión de tareas y **Jira** para la planificación de sprints proporcionará al equipo herramientas robustas y adaptables para monitorear el progreso y gestionar el backlog.

Justificación: La elección de **Trello** y **Jira** [12] permitirá al equipo trabajar de manera más eficiente, facilitando la gestión de tareas y el seguimiento del progreso, alineado con la metodología ágil adoptada.

Conclusión: El análisis de las alternativas y la selección de las herramientas y metodologías adecuadas son cruciales para el éxito de **MusicNet**. La combinación de enfoques ágiles y tecnologías específicas permitirá al equipo abordar los desafíos del proyecto y desarrollar una plataforma que no solo cumpla con las expectativas de los usuarios, sino que también ofrezca una experiencia de enseñanza musical superior.

8.2 Lenguajes y Herramientas

Para el desarrollo de **MusicNet**, se utilizarán varios lenguajes de programación y herramientas tecnológicas que son fundamentales para asegurar la eficacia y el rendimiento del sistema. A continuación, se presenta un análisis de los lenguajes y herramientas seleccionadas, así como su justificación.

- **JavaScript**: [21] Es el principal lenguaje de programación que se utilizará en el desarrollo de **MusicNet**. Este lenguaje es ideal para la creación de aplicaciones en tiempo real y su compatibilidad con bibliotecas como **AubioJS** y **Tone.js** lo convierte en una opción excelente para el procesamiento de audio.

- **HTML5 y CSS3**: Estos lenguajes se utilizarán para la estructura y el diseño de la interfaz de usuario. HTML5 permite la creación de elementos multimedia y el soporte de audio, mientras que CSS3 se utilizará para mejorar la experiencia visual de la aplicación, haciendo que la interfaz sea atractiva y fácil de usar.

Justificación: El uso de JavaScript, junto con HTML5 y CSS3, proporciona una base sólida para el desarrollo de aplicaciones web interactivas y responsivas. Estas tecnologías son ampliamente utilizadas en la industria y cuentan con una gran comunidad de soporte, facilitando el desarrollo y la resolución de problemas.

- **AubioJS**: Esta biblioteca será crucial para la detección de notas y la sincronización de sonidos en tiempo real. AubioJS permite analizar el audio en vivo y extraer información útil, como el tono y el ritmo, lo que es esencial para el funcionamiento de **MusicNet**.
- **Tone.js**: Se utilizará para la síntesis y reproducción de audio dentro de la aplicación. Esta biblioteca permite manejar el audio de forma avanzada y ofrece capacidades de creación musical en tiempo real, lo que enriquece la experiencia de los usuarios.

Justificación: La integración de **AubioJS** y **Tone.js** no solo facilita la implementación de características esenciales para la plataforma, sino que también asegura que **MusicNet** sea capaz de proporcionar una experiencia de usuario atractiva y de alta calidad en términos de audio.

- **Trello**: Utilizado para la gestión de tareas, Trello permite organizar el trabajo del equipo mediante tableros Kanban, facilitando la visualización del estado de las tareas y el flujo de trabajo.
- **Jira**: Será la herramienta principal para la gestión de sprints y seguimiento del progreso del proyecto. Permite crear historias de usuario, gestionar el backlog y realizar seguimiento de problemas.
- **Git**: Este sistema de control de versiones será fundamental para gestionar el código fuente del proyecto. Permite al equipo trabajar de manera colaborativa, manteniendo un historial de cambios y facilitando la gestión de ramas. [6]

Justificación: El uso de estas herramientas proporciona una estructura organizada para el desarrollo del proyecto, permitiendo una colaboración efectiva y la gestión eficiente de tareas y recursos.

8.2.1 Análisis de Alternativas y Justificación

Al evaluar las opciones de lenguajes y herramientas para el desarrollo de **MusicNet**, se consideraron varias alternativas. A continuación, se presentan las opciones analizadas y las justificaciones para la selección final.

- **Alternativa 1 Python**: Aunque Python es un lenguaje poderoso y

ampliamente utilizado en el ámbito de la IA y el procesamiento de datos, no es ideal para el desarrollo de aplicaciones web en tiempo real, ya que generalmente requiere un framework adicional (como Django o Flask) para manejar la funcionalidad del frontend.
- **Alternativa 2 JavaScript**: JavaScript es el lenguaje más adecuado para el desarrollo de aplicaciones web interactivas, ya que permite la creación de aplicaciones que funcionan directamente en el navegador. La compatibilidad con bibliotecas y frameworks de audio también es superior en JavaScript.

Justificación: La elección de JavaScript, junto con HTML5 y CSS3, permite desarrollar una plataforma robusta y flexible que es compatible con las necesidades de interacción en tiempo real de **MusicNet**.

- **Alternativa 1 Web Audio API**: Aunque la API de audio de la web permite el procesamiento de audio en el navegador, su implementación puede ser más compleja y requerir más trabajo manual en comparación con bibliotecas especializadas como **Tone.js**. Esto podría aumentar el tiempo de desarrollo y la complejidad del código.
- **Alternativa 2 AubioJS y Tone.js**: Ambas bibliotecas son específicas para la detección y la síntesis de audio. Proporcionan una solución más inmediata y sencilla, permitiendo al equipo centrarse en el desarrollo de la funcionalidad del sistema sin complicaciones adicionales.

Justificación: La selección de **AubioJS** y **Tone.js** se basa en su facilidad de uso, la documentación extensa y la capacidad de integración directa en aplicaciones web, lo que acelera el desarrollo y mejora la calidad del producto final.

- **Alternativa 1 Asana**: Asana es otra herramienta de gestión de proyectos que permite la organización de tareas. Sin embargo, su enfoque no es tan adaptable a metodologías ágiles como **Trello** y **Jira**, lo que podría limitar la capacidad del equipo para responder a los cambios en el proyecto.
- **Alternativa 2 Trello y Jira**: Ambas herramientas son ampliamente reconocidas en la comunidad de desarrollo ágil y permiten una gestión eficiente del trabajo en equipo, facilitando la colaboración y la comunicación.

Justificación: La elección de **Trello** para la gestión de tareas y **Jira** para la planificación de sprints proporciona al equipo una base sólida para trabajar de manera colaborativa y mantener la transparencia en el progreso del proyecto.

8.3 Plan de Aceptación del Producto

El **Plan de Aceptación del Producto** es una parte crítica del desarrollo de **MusicNet**, ya que establece los criterios y procedimientos necesarios para evaluar si la plataforma cumple con las expectativas y requisitos de los usuarios finales y

los directores del proyecto. A continuación, se detallan los criterios de aceptación y el proceso de evaluación.

Criterios de Aceptación:

1. **Funcionalidad Principal**:
    - El sistema debe permitir a los usuarios tocar el piano en un entorno multijugador, donde la sincronización de notas entre jugadores debe ser precisa con un margen de error no superior a 50 ms.
    - La plataforma debe proporcionar retroalimentación instantánea sobre la precisión de las notas tocadas, utilizando **AubioJS** para detectar sonidos en tiempo real.
2. **Calidad de Audio**:
    - La calidad de audio transmitido debe ser clara y sin distorsiones, con latencia máxima permitida de 100 ms. El sistema debe ser capaz de manejar hasta un 20% de pérdida de paquetes sin afectar significativamente la experiencia del usuario.
    - La compensación de pérdida de paquetes debe ser efectiva, asegurando que el audio no se interrumpa o degrade durante la transmisión.
3. **Experiencia del Usuario**:
    - La interfaz gráfica debe ser intuitiva y fácil de navegar, con elementos visuales claros que guíen a los usuarios a través de sus lecciones y ejercicios.
    - Se realizarán pruebas de usabilidad con usuarios finales para obtener retroalimentación sobre la experiencia general de la plataforma.
4. **Documentación**:
    - Se debe proporcionar documentación técnica completa y un manual de usuario accesible que explique todas las funciones de la plataforma, así como ejemplos prácticos de uso.

Proceso de Evaluación:

1. **Pruebas Funcionales**:
    - Se llevarán a cabo pruebas unitarias y de integración para asegurar que todas las funcionalidades estén operativas. Cada módulo será evaluado de manera independiente antes de la integración final.

2. **Pruebas de Rendimiento**:

   - Se realizarán pruebas de rendimiento en redes con diferentes niveles de latencia y pérdida de paquetes para asegurar que la plataforma funcione adecuadamente en condiciones reales.

3. **Revisiones de Usuario**:

   - Se invitará a un grupo de usuarios finales (estudiantes y docentes) a participar en sesiones de prueba. Durante estas sesiones, se recopilará retroalimentación sobre la usabilidad y la funcionalidad de la plataforma.

   - Las sesiones incluirán tareas específicas que los usuarios deberán completar, y se recogerán comentarios sobre la experiencia general y cualquier problema encontrado.

4. **Informe de Aceptación**:

   - Al finalizar todas las pruebas, se elaborará un informe de aceptación que resuma los resultados de las pruebas funcionales, de rendimiento y las revisiones de los usuarios. Este informe se presentará a los directores del proyecto para su evaluación final.

8.4 Organización del Proyecto y Comunicación

8.4.1 Interfaces Externas

Entidad	Descripción	Responsabilidades
Usuarios Finales	Estudiantes y docentes que interactúan con la plataforma MusicNet.	Acceso a la plataforma a través de un navegador web. Tocar el piano y competir o colaborar en tiempo real. Acceso a manuales y tutoriales.
Directores del Proyecto	Equipo responsable de supervisar el desarrollo de MusicNet.	Revisión del progreso del desarrollo. Acceso a un dashboard con resultados de pruebas. Recepción de informes periódicos sobre pruebas funcionales y de rendimiento.

Tabla 4.interfaces externas

8.4.2 Organigrama y Descripción de Roles

8.4.2.1 Organigrama del Proyecto

El equipo de **MusicNet** está conformado por dos directores de proyecto con experiencia en ingeniería de sistemas y electrónica, y dos estudiantes con especialidades complementarias en desarrollo de software multijugador y redes de telecomunicaciones. A continuación, se presenta el organigrama del proyecto con la distribución de roles:

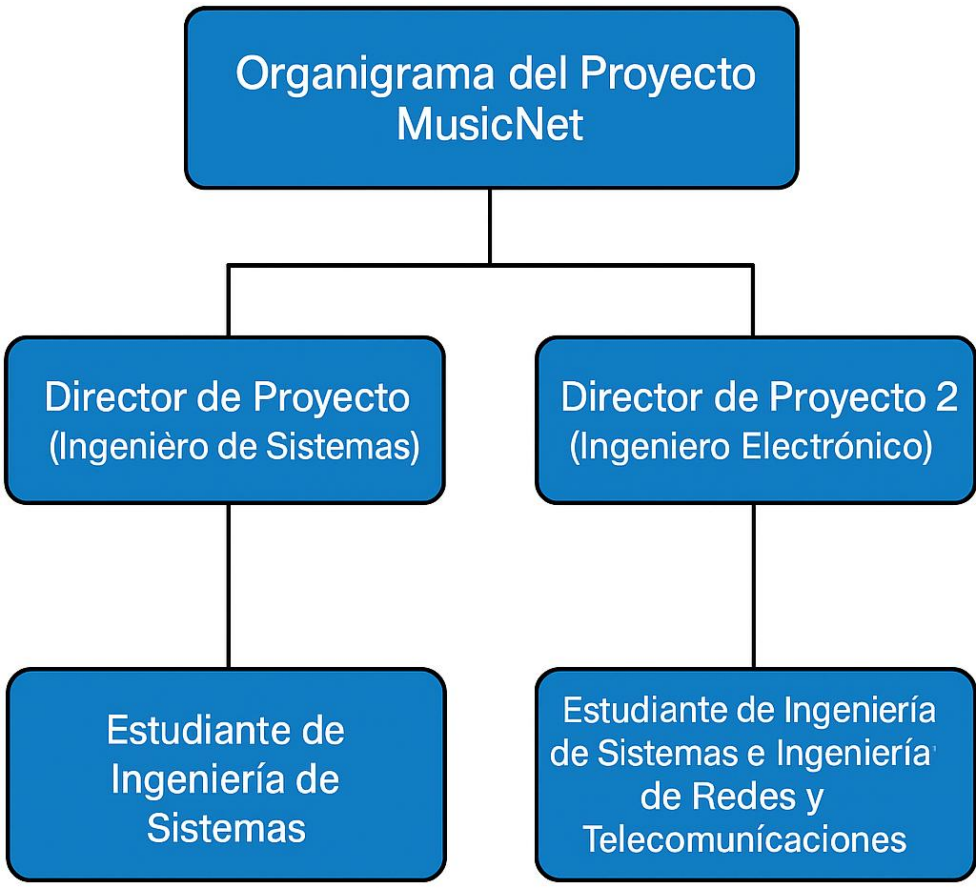

Ilustración 2. Organigrama

8.4.2.2 Descripción de Roles

1. Director de Proyecto 1 (Ingeniero de Sistemas):

    - Responsabilidades: Liderar la organización y ejecución general del proyecto, coordinar las actividades del equipo y garantizar el cumplimiento de los objetivos técnicos, especialmente en el desarrollo del sistema y sus funcionalidades.
    - Perfil**:** *Rafael Vicente Páez Méndez, PhD.*
      Formación: Ingeniería de Sistemas (U. Católica), Especialización en Seguridad de Redes Informáticas, Doctorado en Ingeniería Telemática (UPC, España).
      Áreas de investigación: Seguridad digital, redes informáticas, sistemas distribuidos, IDS.

2. Director de Proyecto 2 (Ingeniero Electrónico):

    - Responsabilidades: Supervisar los aspectos técnicos relacionados con la transmisión de datos, procesamiento de señales y latencia. Brindar apoyo interdisciplinario con base en conocimientos de tecnología musical.
    - Perfil**:** *Gustavo Ramírez-Espinosa, PhD.*

Formación: Ingeniero Electrónico y M.Sc. en Electrónica (PUJ), Doctorado en Ingeniería Informática y Control (Politécnico di Torino, Italia).
Áreas de interés: IoT, Tecnología Musical, sistemas embebidos, ciberseguridad, redes.
Contribuciones destacadas a los ODS: Educación de calidad (ODS 4) y comunidades sostenibles (ODS 11).

3. Estudiante de Ingeniería de Sistemas:

    - Responsabilidades: Desarrollo completo del módulo multijugador de la plataforma, incluyendo la implementación de la conexión entre jugadores, sincronización en tiempo real usando WebRTC y lógica de juego colaborativo. Encargada del backend interactivo y de la integración con la interfaz visual.
    - Perfil: Estudiante con habilidades avanzadas en desarrollo web, lógica de sincronización en tiempo real, WebRTC, y gestión de la interfaz con librerías como Phaser y tecnologías frontend modernas.

4. Estudiante de Ingeniería de Sistemas e Ingeniería de Redes y Telecomunicaciones:

    - Responsabilidades: Supervisar y configurar los aspectos de red, incluyendo la implementación de estrategias para la compensación de pérdida de paquetes, medición de latencia y realización de pruebas de rendimiento en condiciones de red variables. Encargada del monitoreo y mejora de la calidad de la transmisión en tiempo real.
    - Perfil: Estudiante con experiencia sólida en redes y telecomunicaciones, especializada en gestión de latencia, implementación de modelos predictivos (como PARCNet) y pruebas de estabilidad y rendimiento en entornos inestables.

Cada miembro del equipo tiene asignadas responsabilidades específicas, permitiendo una distribución clara del trabajo y contribuyendo al desarrollo exitoso de MusicNet. Los directores guiarán a los estudiantes y supervisarán el progreso del proyecto, asegurando que cada componente se desarrolle de acuerdo con los objetivos y requisitos definidos

Administración del Proyecto

9.1 Métodos y Herramientas de Estimación

La administración efectiva del proyecto **MusicNet** depende en gran medida de la precisión en las estimaciones de tiempo y recursos. Para lograr esto, se utilizarán una combinación de métodos ágiles de estimación, priorizando la adaptabilidad y el seguimiento cercano al progreso real. Dado que el equipo es reducido y compuesto por estudiantes con supervisión docente, las herramientas y métodos se aplicarán de manera práctica y contextualizada.

Métodos de Estimación:

1. **Estimación por Puntos de Historia**: Este método se utiliza en el marco de Scrum, donde las tareas se asignan un puntaje en función de su complejidad y esfuerzo. Los puntos de historia son una medida relativa que permite al equipo evaluar la carga de trabajo y estimar cuántas tareas pueden completar en un sprint.
2. **Estimación Basada en Experiencia**: El equipo también utilizará la experiencia previa en proyectos similares para estimar las tareas. Esto incluye revisar tiempos de entrega y resultados de proyectos anteriores que involucren tecnologías y metodologías similares. La experiencia del equipo proporcionará una base sólida para la estimación de tareas y ayudará a ajustar las expectativas a medida que avanza el proyecto.
3. **Descomposición de Tareas**: Cada tarea del proyecto se descompondrá en actividades más pequeñas y manejables, permitiendo estimaciones más precisas. Al dividir las tareas complejas en partes más simples, se facilita la identificación de los recursos necesarios y el tiempo que tomará completarlas.
4. **Ajuste Continúo** Las estimaciones serán dinámicas y se ajustarán en función de los descubrimientos técnicos. Esté enfoque permitirá enfocar el esfuerzo en resolver problemas críticos a medida que surgían.

Herramientas de Estimación:

1. **Trello**: Será utilizado como una herramienta visual para la gestión del trabajo, donde se podrán organizar las tareas en diferentes listas que representen el estado actual (pendientes, en progreso, completadas). Esto permitirá al equipo tener una visión clara del flujo de trabajo y facilitar la priorización de las tareas.[22]
2. **Jira**: Se utilizará para la planificación de sprints y seguimiento del backlog. Se crearán historias de usuario con descripciones detalladas y estimaciones de puntos de historia. Jira permite el seguimiento del progreso de cada tarea y proporciona informes sobre la velocidad del equipo y el tiempo de ciclo. Referencias[12]

Proceso de Estimación:

1. **Revisión de Requerimientos**: Antes de realizar estimaciones, se revisarán todos los requerimientos del proyecto. Esto incluye la identificación de las funcionalidades clave y la priorización de las tareas en función de la importancia para el éxito del proyecto.
2. **Establecimiento de Tareas**: A partir de los requerimientos revisados, se descompondrán en tareas específicas. Cada tarea se documentará con su descripción, puntos de historia y responsables asignados.
3. **Planificación y Priorización**: Se utilizará Trello para documentar y organizar las tareas, agrupándolas en columnas según su estado (pendiente, en proceso, completada). Aunque no se definirán sprints formales, se aplicaran principios de priorización y organización visual que permitieron tener claridad sobre el avance.

4. **Estimación Colaborativa Basada en Experiencia:** Las tareas inicialmente serán estimadas en reuniones informales, considerando la experiencia del equipo y la complejidad técnica percibida. No se asignarán puntos de historia en todos los casos, pero se hará una evaluación relativa del esfuerzo requerido para cada tarea
5. **Ajuste Continuo**: A medida que avanza el proyecto, se realizarán revisiones periódicas de las estimaciones. Si se detectan desviaciones significativas, se ajustarán las estimaciones en función del rendimiento real del equipo y los resultados obtenidos en los sprints anteriores.

9.2 Inicio del proyecto

9.2.1 Entrenamiento del Personal

El entrenamiento del personal es fundamental para garantizar que todos los miembros del equipo tengan las habilidades y conocimientos necesarios para llevar a cabo sus responsabilidades en el proyecto **MusicNet**. El entrenamiento se dividirá en varias sesiones, centradas en las tecnologías, metodologías y herramientas que se utilizarán durante el desarrollo.

Plan de Entrenamiento:

1. **Introducción a la Metodología Ágil**: Se realizarán sesiones de capacitación sobre las metodologías ágiles, específicamente sobre **Scrum** y **Kanban**. El equipo aprenderá sobre los principios de trabajo ágil, la planificación de sprints, la gestión del backlog y las reuniones diarias.
2. **Capacitación Técnica**: Los miembros del equipo recibirán capacitación específica en las herramientas y tecnologías que utilizarán, como **AubioJS** para la detección de sonido y **Tone.js** para la síntesis de audio. Esto incluirá ejercicios prácticos para asegurar que el equipo se sienta cómodo utilizando estas herramientas en el desarrollo diario.
3. **Revisión de Mejores Prácticas**: Se presentarán mejores prácticas en el desarrollo de software y gestión de proyectos, centrándose en la colaboración, la comunicación y la documentación adecuada. Esto ayudará a establecer un ambiente de trabajo colaborativo y productivo.
4. **Simulaciones de Pruebas**: Se llevarán a cabo simulaciones de pruebas para familiarizar al equipo con el proceso de aseguramiento de calidad y las herramientas que se utilizarán para las pruebas de funcionalidad y rendimiento.

Objetivo del Entrenamiento:

Asegurar que todos los miembros del equipo comprendan las metodologías ágiles, las tecnologías utilizadas y los procedimientos de calidad, creando un equipo cohesionado y preparado para abordar los desafíos del proyecto.

9.2.2 Infraestructura

La infraestructura tecnológica es un componente crítico para el desarrollo y funcionamiento de **MusicNet**. Asegurar que todos los recursos necesarios estén

disponibles y configurados adecuadamente facilitará un flujo de trabajo eficiente y la colaboración efectiva entre los miembros del equipo.

Componentes de la Infraestructura:

1. **Entorno de Desarrollo**:
    - Cada miembro del equipo contará con un entorno de desarrollo local configurado en su computadora, incluyendo:
        - **Servidor HTTP local con Python**: Se utilizará el comando Python -m http. server para servir los archivos del proyecto localmente durante el desarrollo. Esta solución es simple, no requiere dependencias adicionales y es adecuada para aplicaciones web que se ejecutan completamente en el navegador.
        - **Editores de Código**: Se recomendará el uso de **Visual Studio Code** por su facilidad de uso y las extensiones disponibles para JavaScript.
        - **Sistema de Control de Versiones (Git)**: Se configurará un repositorio en **GitHub** para gestionar el código fuente y las colaboraciones del equipo.
2. **Herramientas de Comunicación y Gestión**:
    - **Trello**: Para gestionar el flujo de trabajo y las tareas, permitiendo a los miembros del equipo visualizar el estado de cada tarea de manera clara.
    - **Jira**: Para la planificación de sprints y seguimiento del progreso, con capacidad para crear historias de usuario y gestionar el backlog.
3. **Redes de Prueba**:
    - Se establecerán entornos de red de prueba para simular diferentes condiciones de latencia y pérdida de paquetes. Esto incluirá el uso de herramientas como **Clumsy** y **NetEm**, que permitirán controlar y modificar las condiciones de la red para evaluar el rendimiento de **MusicNet** en situaciones reales.
4. **Infraestructura de Servicios en Línea:**
    - **Servidor de Señalización WebRTC**: Desplegado en Render.com, encargado de la conexión entre pares para la sincronización en tiempo real.
    - **API de PARCNet**: También alojada en Render.com, encargada del manejo predictivo de datos perdidos mediante modelos de IA.
    - **Despliegue de la Aplicación**: La versión final del cliente será desplegada mediante Surge.sh, permitiendo el acceso público al sistema

desde navegadores web sin necesidad de backend dedicado para contenido estático.

Justificación de la Infraestructura:
La infraestructura está diseñada para ser ligera, accesible y efectiva. Se evitan soluciones complejas en la nube, optando por servicios gratuitos y escalables como Render.com y Surge.sh. Esta estrategia permite un desarrollo ágil y pruebas realistas sin incurrir en costos innecesarios ni depender de entornos corporativos complejos.

9.3 Planes de Trabajo del Proyecto

9.3.1 Descomposición de Actividades

Para una gestión eficiente del proyecto, es esencial descomponer las actividades en tareas más pequeñas y manejables. La descomposición de actividades facilita la planificación, asignación de recursos y seguimiento del progreso. A continuación, se presenta un desglose de las principales actividades del proyecto **MusicNet**:

1. **Análisis de Requisitos**:
    - Reuniones iniciales con los directores para recopilar requerimientos.
    - Documentación de los requisitos funcionales y no funcionales.
2. **Diseño de Arquitectura e Interfaz**:
    - Creación de diagramas de flujo y arquitectura del sistema.
    - Diseño de prototipos de la interfaz gráfica utilizando herramientas de diseño como **Figma**.
3. **Desarrollo del Sistema**:
    - Implementación de interfaz.
    - Integración de **AubioJS** para la detección de sonidos.
    - Desarrollo de la lógica multijugador y sincronización.
4. **Compensación de Pérdida de Paquetes**:
    - Implementación de modelos de IA para la compensación de pérdida de paquetes.
    - Ajuste del sistema para manejar condiciones de red inestables.
5. **Pruebas e Integración**:
    - Realización de pruebas unitarias y de integración.
    - Pruebas de rendimiento en diferentes escenarios de red.
6. **Documentación y Entrenamiento**:
    - Redacción de la documentación técnica y manuales de usuario.

- Entrenamiento del personal y usuarios finales sobre el uso de la plataforma.

9.3.2 Calendarización

La calendarización del proyecto **MusicNet** se estructurará en **sprints** semanales, lo que permitirá al equipo trabajar de manera iterativa y ajustarse a los comentarios y resultados de las pruebas. A continuación, se presenta un cronograma general de las fases del proyecto.

Fase	Duración	Descripción
Análisis de Requisitos	1 semana	Revisión y documentación de requerimientos funcionales y no funcionales.
Diseño de Arquitectura e Interfaz	2 semanas	Diseño de la arquitectura del sistema y prototipos de interfaz gráfica.
Desarrollo del Sistema	7 semanas	Implementación de todas las funcionalidades clave: backend, frontend, detección de sonidos y lógica multijugador.
Compensación de perdida de paquetes	2 semanas	Integración y pruebas de los modelos de IA para compensar la pérdida de paquetes.
Pruebas e integración	2 semanas	Pruebas unitarias, de integración y de rendimiento en condiciones de red inestables.
Documentación	1 semana	Preparación de la documentación final
Revisión final y cierre	1 semana	Revisión de los entregables finales y cierre del proyecto.

Tabla 5. Calendarización

Metodología de Revisión:

- Al finalizar cada fase, se llevará a cabo una reunión de revisión para evaluar el progreso, discutir los logros y planificar las siguientes actividades.
- Las reuniones de retroalimentación continuarán a lo largo de cada sprint, permitiendo ajustes en función de los resultados de las pruebas y las revisiones de los usuarios.

9.3.3 Asignación de Recursos

La asignación de recursos es crucial para garantizar el éxito del proyecto MusicNet. A continuación, se describen los recursos clave que se requerirán y cómo se asignarán a lo largo del proyecto:

Recursos Humanos:

- **Desarrolladores:** desarrolladores, responsables de la implementación de las funcionalidades del sistema. Se asignarán a tareas específicas basadas en su experiencia y habilidades.

Recursos Tecnológicos:

- **Herramientas de Gestión:** Se asignarán licencias para herramientas de gestión de proyectos como Trello y Jira, así como herramientas de control de versiones como Git.

Justificación de la Asignación de Recursos: La correcta asignación de recursos humanos y tecnológicos es esencial para garantizar que MusicNet se desarrolle según lo planificado y que se cumplan todos los objetivos del proyecto. El equipo debe estar compuesto por profesionales con la experiencia adecuada para abordar los desafíos técnicos y garantizar un desarrollo fluido.

Monitoreo y Control del Proyecto

10.1 Administración de Requerimientos

La administración de requerimientos es un aspecto crítico del proyecto **MusicNet**, ya que garantiza que se satisfagan las expectativas y necesidades de los usuarios finales. Este proceso se llevará a cabo de forma continua durante todo el ciclo de vida del proyecto, asegurando que cualquier cambio o adición se gestione adecuadamente.

Proceso de Administración de Requerimientos:

1. **Recolección de Requerimientos**:
    - Se realizarán reuniones iniciales con los directores del proyecto y usuarios finales para identificar las necesidades específicas y expectativas sobre **MusicNet**. Durante estas reuniones, se discutirán las funcionalidades deseadas, los problemas existentes en otras plataformas y las expectativas de rendimiento.
    - Se utilizarán encuestas y entrevistas para obtener retroalimentación directa de los estudiantes y docentes, ayudando a definir los requerimientos funcionales y no funcionales.
2. **Documentación de Requerimientos**:
    - Todos los requerimientos identificados serán documentados en un formato claro y accesible, detallando las características funcionales del sistema y las expectativas de calidad. Se crearán documentos de especificación de requisitos que incluirán descripciones detalladas y criterios de aceptación para cada funcionalidad.
    - Esta documentación se mantendrá actualizada a medida que se realicen cambios y ajustes en los requerimientos.
3. **Priorizar Requerimientos**:
    - Los requerimientos se clasificarán en función de su importancia y urgencia. Se utilizará la metodología **MoSCoW** (Must have, Should have, Could have, Won't have) para priorizar los requisitos, asegurando que se aborden primero aquellos que son críticos para el

funcionamiento del sistema.
   - El equipo revisará las prioridades en cada reunión de planificación de sprints, permitiendo ajustar el enfoque según las necesidades emergentes del proyecto.

4. **Gestión de Cambios**:

   - Cualquier cambio en los requerimientos será gestionado mediante un proceso formal de gestión de cambios. Esto incluirá la documentación del cambio propuesto, la evaluación de su impacto en el tiempo y el presupuesto, y la aprobación de los directores del proyecto antes de su implementación.
   - Se establecerá un sistema de tickets para rastrear y documentar los cambios solicitados y su estado.

5. **Validación de Requerimientos**:

   - Se realizarán revisiones periódicas de los requerimientos con los directores y usuarios clave para asegurar que se mantengan alineados con las expectativas y necesidades del proyecto. Esto incluirá demostraciones de las funcionalidades implementadas y sesiones de feedback.
   - Al final de cada sprint, se evaluará si se han cumplido los criterios de aceptación para cada requerimiento implementado, asegurando que la funcionalidad esté validada antes de continuar con el desarrollo.

10.2 Monitoreo y Control de Progreso

El monitoreo y control del progreso del proyecto es esencial para asegurar que **MusicNet** se desarrolle de acuerdo con el cronograma y dentro del presupuesto establecido. A continuación, se describen las herramientas, métodos y métricas que se utilizarán para supervisar el avance del proyecto.

Herramientas de Monitoreo:

1. **Jira**:

   - Se utilizará **Jira** para gestionar el backlog del proyecto y realizar el seguimiento del progreso de las tareas. Cada tarea se representará como una historia de usuario en Jira, permitiendo al equipo visualizar el estado actual y priorizar el trabajo en función de la importancia.
   - Las métricas de rendimiento, como la velocidad del equipo y el tiempo de ciclo, se rastrearán en Jira para evaluar el progreso y la eficiencia del equipo a lo largo del desarrollo.

2. **Trello**:

   - **Trello** será utilizado como un tablero visual para el seguimiento de las tareas en curso. Cada tarjeta en el tablero representará una tarea

específica, y los miembros del equipo podrán mover las tarjetas entre columnas para indicar su estado (pendiente, en progreso, completada).
    - Este enfoque visual permite una rápida identificación de cuellos de botella y ayuda a gestionar el flujo de trabajo del equipo.

Métricas de Progreso:

1. **Velocidad del Equipo**:
    - Se medirá cuántos puntos de historia se completan en cada sprint. La velocidad se calculará al final de cada sprint, y se utilizará para prever el trabajo que se puede completar en futuros sprints.
    - Esta métrica ayudará a evaluar la capacidad del equipo y a ajustar las expectativas en función del rendimiento real.
2. **Tiempo de Ciclo**:
    - El tiempo de ciclo se calculará desde que una tarea se inicia hasta que se completa. Esta métrica permitirá al equipo identificar cuán eficientemente se están completando las tareas y ayudará a identificar áreas donde se puedan mejorar los procesos.
3. **Pruebas y Validaciones**:
    - La cantidad de errores encontrados durante las pruebas unitarias y de integración será monitoreada. Esto ayudará a asegurar que se mantenga un alto estándar de calidad a lo largo del desarrollo y a identificar patrones en la aparición de errores.
    - Se realizarán informes semanales que resuman los hallazgos de las pruebas, y se asignarán tareas específicas para resolver los problemas identificados.
4. **Feedback de Usuarios**:
    - Se recogerá feedback de los usuarios finales (estudiantes y docentes) durante las pruebas y sesiones de revisión. Esto incluirá encuestas y entrevistas que evalúen su satisfacción con la plataforma y la facilidad de uso.
    - El feedback será analizado y se utilizará para realizar ajustes en las funcionalidades y mejorar la experiencia general de los usuarios.

Revisiones Periódicas:

- Se establecerán reuniones de revisión de progreso cada semana, al final de cada sprint, donde el equipo discutirá el progreso alcanzado, las tareas completadas y cualquier obstáculo encontrado.
- Durante estas reuniones, se revisarán las métricas de rendimiento y se tomarán decisiones sobre ajustes necesarios en la planificación, el enfoque del

trabajo y las prioridades de las tareas.

10.3 Cierre del Proyecto

El cierre del proyecto **MusicNet** es una fase crucial que garantiza que todos los entregables se hayan completado y que el producto final cumpla con los requisitos establecidos. Este proceso incluirá varias actividades clave:

1. **Revisión Final**:
    - Se llevará a cabo una revisión final del sistema para asegurarse de que todas las funcionalidades estén implementadas y funcionen como se esperaba. Esto incluirá la verificación de que todos los criterios de aceptación se hayan cumplido.
2. **Documentación**:
    - Se completará la documentación técnica y el manual del usuario. Estos documentos se entregarán a los directores del proyecto y se pondrán a disposición de los usuarios finales.
3. **Entrega Formal**:
    - Una vez que se haya completado la revisión y la documentación, se realizará una entrega formal del producto final a los directores del proyecto. Esto incluirá una presentación que muestre las funcionalidades de la plataforma y cómo utilizarla.

Entrega del Producto

La entrega del producto **MusicNet** se realizará de manera estructurada y organizada, garantizando que todos los componentes del sistema estén completos y funcionando según lo previsto.

1. **Fase de Preparación**: Antes de la entrega, se llevarán a cabo pruebas finales para garantizar que todas las funcionalidades estén operativas y se cumplan los criterios de aceptación.
2. **Documentación**: Se entregarán todos los documentos relacionados con el proyecto, incluyendo la documentación técnica y el manual del usuario. Estos documentos proporcionarán instrucciones claras sobre cómo utilizar la plataforma y solucionarán problemas comunes.
3. **Presentación Final**: Se organizará una presentación final donde se demostrará el funcionamiento de **MusicNet** a los directores y otros interesados. Durante esta presentación, se mostrarán las características clave de la plataforma y se responderán preguntas sobre su uso.
4. **Implementación de Feedback**: Cualquier feedback recibido durante la presentación se documentará y se considerará para futuras actualizaciones y mejoras del sistema. Se establecerá un plan para implementar estos cambios en futuras versiones de **MusicNet**.

Procesos de Soporte

12.1 Ambiente de Trabajo

El ambiente de trabajo para el desarrollo de **MusicNet** será colaborativo y adaptativo, diseñado para facilitar la comunicación y el trabajo en equipo. Las herramientas y metodologías elegidas asegurarán que el equipo pueda interactuar de manera efectiva, optimizando la productividad y la creatividad.

Componentes del Ambiente de Trabajo:

1. **Espacio de Trabajo Físico y Virtual**:
    - Aunque el equipo trabajará mayoritariamente de forma remota, se establecerán reuniones periódicas en un espacio físico para fomentar la cohesión del equipo y realizar sesiones de trabajo colaborativo.
    - Se utilizarán herramientas de videoconferencia como **Microsoft Teams** para facilitar las reuniones regulares y garantizar que todos los miembros del equipo estén alineados en sus objetivos y tareas.
2. **Cultura Colaborativa**:
    - Se fomentará una cultura de colaboración donde todos los miembros del equipo se sientan cómodos compartiendo ideas y proporcionando retroalimentación constructiva. Las reuniones de retrospectiva se utilizarán para discutir las dinámicas del equipo y promover un ambiente positivo.
3. **Uso de Herramientas Digitales**:
    - Las herramientas digitales como **Trello** y **Jira** serán fundamentales para la comunicación y gestión del trabajo.
4. **Documentación Accesible**:
    - La documentación del proyecto se mantendrá en un repositorio centralizado (OneDrive), accesible para todos los miembros del equipo. Esto asegurará que la información esté disponible y actualizada, permitiendo una mejor gestión del conocimiento dentro del equipo.

12.2 Análisis y Administración de Riesgos

Descripción del Plan de Administración de Riesgos

El plan de administración de riesgos para MusicNet está diseñado para identificar, evaluar y mitigar los riesgos de manera efectiva, asegurando que el proyecto se mantenga en el camino correcto. Este proceso incluye la participación de todo el equipo y el uso de herramientas para gestionar y controlar los riesgos a lo largo del ciclo de vida del proyecto.

- Responsable: Los directores del Proyecto serán los principales responsables de la administración de riesgos, en coordinación con los líderes de equipo en cada área

- Actividades:
  - Identificación y documentación de riesgos en reuniones iniciales y de revisión de cada sprint.
  - Evaluación de cada riesgo basado en su probabilidad de ocurrencia e impacto.
  - Desarrollo de estrategias de mitigación para los riesgos clasificados como críticos.
  - Monitoreo constante y actualización de riesgos en cada reunión de revisión del sprint.
  - Herramientas Utilizadas: Jira para el seguimiento de riesgos.

Descripción del Proceso de Identificación de Riesgos

El proceso de identificación de riesgos se estructurará en fases, utilizando un diagrama BPMN (Business Process Model and Notation) para ilustrar los pasos.

1. Inicio del Proyecto: Se realiza una sesión de identificación de riesgos con todos los miembros del equipo, donde se analizan riesgos potenciales en cada área clave del proyecto.
2. Revisión por Sprint: Al final de cada sprint, se realiza una evaluación de riesgos en una reunión de revisión. Se identifican nuevos riesgos o cambios en los riesgos ya conocidos.
3. Evaluación y Clasificación de Riesgos: Cada riesgo identificado se evalúa en términos de probabilidad de ocurrencia e impacto en el proyecto.
4. Actualización y Comunicación: Los riesgos se registran y actualizan en el sistema de seguimiento, asegurando que el equipo esté informado y preparado para responder.

Diagrama BPMN del Proceso de Identificación de Riesgos

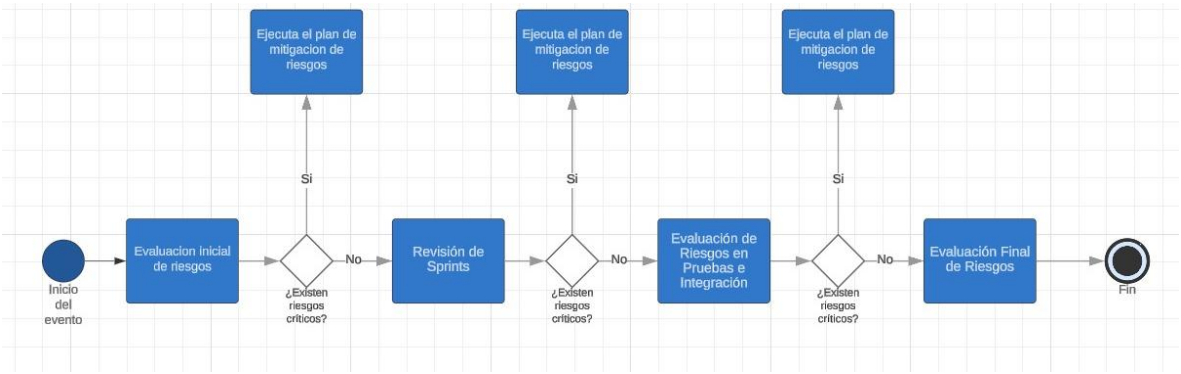

Ilustración 3. BPMN identificación de riesgos

Momentos en el Proyecto para la Evaluación de Riesgos

1. Inicio del Proyecto: Evaluación inicial para identificar riesgos inherentes a la tecnología, recursos y plazos.
2. Revisión de Sprints: Al cierre de cada sprint, revisión de los riesgos actuales y adición de nuevos riesgos según los cambios en el proyecto.
3. Fase de Pruebas e Integración: Evaluación de riesgos específicos relacionados con el rendimiento, la latencia y la sincronización de usuarios.
4. PreEntrega: Evaluación final para garantizar que todos los riesgos críticos se hayan abordado antes de la entrega del producto final.

Riesgos Identificados

La tabla muestra los riesgos del proyecto ordenados por prioridad, calculada en función de su probabilidad e impacto.

Riesgo	Probabilidad	Impacto	Prioridad
Latencia alta	Alta	Alto	1
Pérdida de paquetes	Alta	Alto	2
Complejidad técnica	Media	Alto	3
Retención de usuarios baja	Media	Medio	4
Falta de recursos	Baja	Medio	5

Tabla 6. Riesgos Identificados

Riesgos Críticos con Estrategias de Prevención y Mitigación

Esta tabla presenta los riesgos más importantes del proyecto, indicando las acciones de prevención y mitigación, junto con los responsables y los recursos necesarios para su gestión.

Riesgo	Acción de Prevención	Acción de Mitigación	Recursos Necesarios
Latencia alta	Implementar protocolos de baja latencia desde el inicio	Aplicar IA para compensación de pérdida de paquetes y mejorar sincronización	Herramientas de pruebas de latencia y red
Pérdida de paquetes	Utilizar modelos predictivos de IA para compensación	Reforzar con pruebas de rendimiento en diferentes redes y ajustar el algoritmo de IA si es necesario	Equipo de pruebas de red
Complejidad técnica	Planificación y revisión de diseño con expertos en cada tecnología involucrada	Asignar tareas complejas a especialistas y realizar pruebas iterativas en cada módulo	Tiempo extra en la fase de pruebas y revisión
Retención de usuarios baja	Realizar sesiones de prueba con usuarios y recoger feedback	Mejorar la interfaz y agregar elementos de gamificación para	Herramientas de pruebas de usuario

		aumentar la interacción y satisfacción del usuario	
Falta de recursos	Planificación de contingencias y recursos adicionales antes de la fase crítica del proyecto	Buscar asociaciones con otras instituciones y aumentar la disponibilidad de recursos cuando sea necesario	Acceso a redes de contacto en el sector

Tabla 7. Estrategias Riesgos Críticos

12.3 Administración de Configuración y Documentación

La administración de configuración y documentación es fundamental para asegurar que el desarrollo de MusicNet sea coherente y que el conocimiento del proyecto esté siempre disponible y actualizado para el equipo. A continuación, se presentan los procesos y herramientas que se implementarán para gestionar la configuración y la documentación de forma estructurada.

Administración de Configuración

Ítems de Configuración Identificados

1. Código Fuente: Repositorio de código alojado en GitHub, donde se gestionan todas las contribuciones del equipo. Este repositorio incluye el código para los módulos principales de la plataforma.
2. Políticas de Ramas: Se implementarán políticas de rama para asegurar que cada nueva funcionalidad o corrección se desarrolle en una rama separada. Esto facilita la revisión del código y permite la integración en la rama principal solo después de aprobar las revisiones necesarias.
3. Registro de Cambios (CHANGELOG): Documento que detalla las mejoras y correcciones realizadas en cada versión. Este archivo se actualizará con cada commit significativo y ofrece una visión general del progreso y las modificaciones implementadas.
4. Scripts de Configuración del Entorno: Scripts que permiten la instalación y configuración de dependencias y entornos locales para el equipo. Estos scripts aseguran la consistencia del entorno de desarrollo, evitando incompatibilidades.

Proceso de Control de Cambios a Ítems de Configuración

A continuación, se muestra el flujo del proceso de control de cambios en un diagrama BPMN (Business Process Model and Notation) y su explicación.

Diagrama BPMN

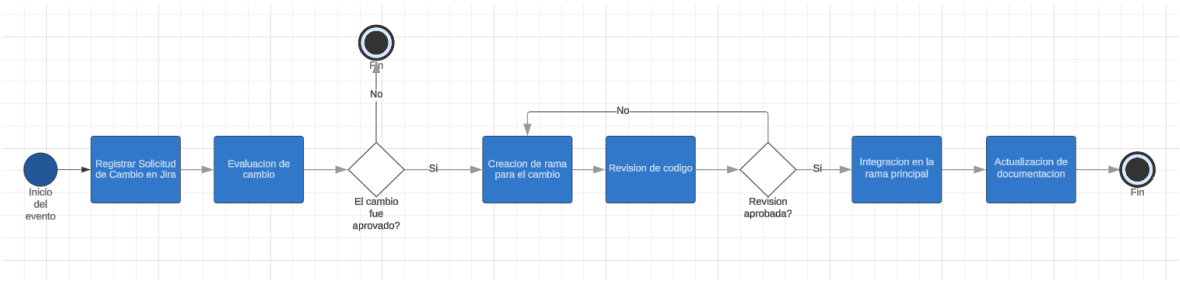

Ilustración 4. BPMN Control de Cambios

1. Solicitud de Cambio: Un miembro del equipo detecta una necesidad de cambio en el código o en otro ítem de configuración y lo registra en Jira como una solicitud de cambio.
2. Evaluación de Cambio: El equipo revisa la solicitud en la próxima reunión de planificación. Se evalúa el impacto del cambio en el cronograma, los recursos y la funcionalidad general del sistema.
3. Creación de Rama para el Cambio: Si el cambio es aprobado, se crea una rama en GitHub específicamente para ese cambio. El desarrollador responsable realiza las modificaciones en esta rama.
4. Revisión de Código: El código actualizado pasa por una revisión para asegurar que cumple con los estándares de calidad y no introduce errores. Las revisiones son realizadas por un compañero de equipo o el desarrollador líder.
5. Integración en la Rama Principal: Tras la aprobación de la revisión, el cambio se fusiona en la rama principal. Se actualiza el CHANGELOG y se envía una notificación al equipo.
6. Actualización de Documentación: Se actualizan los documentos relacionados, incluyendo la documentación técnica y el manual de usuario, si el cambio afecta al comportamiento o configuración de la aplicación.

Documentación

Ítems de Documentación Identificados

1. Documentación Técnica: Describe la arquitectura del sistema, los módulos, las interfaces y las bibliotecas utilizadas. Esta documentación se actualiza continuamente y sirve como referencia para los desarrolladores actuales y futuros.
2. Manual del Usuario: Manual accesible que incluye instrucciones de uso de MusicNet, con secciones detalladas sobre las funcionalidades principales y solución de problemas comunes. Este manual estará disponible para los usuarios finales durante el lanzamiento de la plataforma.
3. Almacenamiento de Documentación: Toda la documentación se almacenará en un repositorio centralizado, OneDrive, al que tendrán acceso todos los miembros del equipo. Esto asegura que la información esté organizada y sea accesible.
4. Revisión de Documentación: Las revisiones de la documentación se

programarán al final de cada sprint para garantizar que sea clara y esté alineada con el estado actual del proyecto. Los miembros del equipo podrán proporcionar retroalimentación para mejorar la claridad y precisión.

Tabla de Artefactos de Documentación y Código

Artefacto	**Descripción**	**Momento de Creación o Refinamiento**
Código Fuente	Implementación de funcionalidades en el repositorio de GitHub	En cada sprint según el plan de desarrollo
Políticas de Ramas	Normas para el uso de ramas en Git	Al inicio del proyecto
Registro de Cambios	Documento que detalla cambios en el código y actualizaciones importantes	Con cada commit significativo
Scripts de Configuración	Scripts para configurar el entorno de desarrollo	Al inicio y se ajustan según necesidades
Documentación Técnica	Detalles de arquitectura, módulos, y bibliotecas	En cada sprint según los cambios en el sistema
Manual del Usuario	Instrucciones de uso y solución de problemas	Completo al finalizar el proyecto
Almacenamiento de Documentación	Repositorio de almacenamiento y gestión de documentos	Inicial y se actualiza continuamente
Revisión de Documentación	Proceso para revisar y actualizar la documentación del proyecto	Al final de cada sprint

Tabla 8. Artefactos Documentación y Código

Este plan de administración de configuración y documentación asegura que MusicNet se desarrolle de forma estructurada, con registros claros de los cambios y documentación actualizada para guiar al equipo en cada fase del proyecto.

12.4 Métricas y Proceso de Medición

La implementación de métricas y un proceso de medición efectivos es esencial para evaluar el progreso del proyecto **MusicNet** y garantizar que se cumplan los objetivos establecidos. A continuación, se presentan las métricas clave que se utilizarán para medir el éxito del proyecto.

Métricas Clave:

1. **Velocidad del Equipo**: Medida en puntos de historia, la velocidad del equipo se calculará al final de cada sprint. Esta métrica proporcionará una visión clara de la capacidad del equipo para completar tareas y ayudará a ajustar las estimaciones en futuros sprints.
2. **Tasa de Errores**: Se registrarán los errores encontrados durante las pruebas unitarias, de integración y de aceptación. Esta métrica ayudará a evaluar la calidad del código y la efectividad de las pruebas realizadas, permitiendo identificar áreas de mejora.

3. **Cumplimiento de Plazos**: Se monitorizará el porcentaje de tareas completadas en cada sprint en comparación con el plan original. Esta métrica permitirá evaluar la capacidad del equipo para cumplir con los plazos establecidos y facilitará la identificación de posibles cuellos de botella en el proceso de desarrollo.
4. **Satisfacción del Usuario**: Se realizarán encuestas de satisfacción del usuario después de cada prueba y entrega del sistema. La retroalimentación de los usuarios finales proporcionará información valiosa sobre la experiencia de uso y ayudará a identificar áreas que requieran mejoras.

Proceso de Medición:

1. **Recolección de Datos**: Los datos para cada métrica se recopilarán de manera sistemática durante el desarrollo del proyecto. Esto incluirá la revisión de las historias de usuario en **Jira**, el seguimiento de las tareas en **Trello** y la recolección de feedback de los usuarios a través de encuestas.
2. **Análisis de Datos**: Se realizarán análisis regulares de los datos recopilados para evaluar el progreso del proyecto y tomar decisiones basadas en evidencia. Esto incluirá la comparación de las métricas actuales con los objetivos establecidos al inicio del proyecto.
3. **Revisiones Periódicas**: Durante las reuniones de revisión, se presentarán las métricas al equipo y a los directores del proyecto. Esto permitirá discutir el progreso y ajustar las prioridades y enfoques según sea necesario.
4. **Ajustes Basados en Datos**: Si se identifican desviaciones significativas en las métricas, se tomarán decisiones informadas para ajustar el enfoque del proyecto, reasignar recursos o modificar el cronograma.

12.5 Control de Calidad

El control de calidad es un aspecto crítico del desarrollo de **MusicNet**, asegurando que la plataforma cumpla con los estándares de calidad establecidos y que los usuarios tengan una experiencia óptima al utilizarla. A continuación, se describen las estrategias y procesos que se implementarán para garantizar la calidad del producto final.

Estrategias de Control de Calidad:

1. **Pruebas Unitarias**:

   - Se implementarán pruebas unitarias para cada módulo y funcionalidad del sistema. Esto garantizará que cada componente funcione correctamente de manera independiente y que cumpla con los requerimientos especificados.
   - Las pruebas unitarias se automatizarán utilizando frameworks de pruebas, permitiendo que se ejecuten de manera regular y facilitando la detección temprana de errores.

2. **Pruebas de Integración**:

- Las pruebas de integración se llevarán a cabo para verificar que los diferentes módulos del sistema funcionen correctamente juntos. Esto incluirá la evaluación de la interacción entre el backend y el frontend, así como la integración de bibliotecas externas como **AubioJS** y **Tone.js**.
- Se crearán escenarios de prueba que reflejen situaciones del mundo real, asegurando que la plataforma funcione correctamente bajo condiciones normales de uso.

3. **Pruebas de Rendimiento**:

   - Se realizarán pruebas de rendimiento en condiciones de red variables para evaluar la estabilidad y la capacidad de respuesta del sistema. Esto incluirá pruebas bajo diferentes niveles de latencia y pérdida de paquetes, tanto antes como después de integrar el sistema de compensación de pérdida. Esto permitirá validar que la plataforma mantenga una experiencia de usuario de calidad en condiciones de red adversas.
   - Las pruebas de rendimiento se llevarán a cabo utilizando herramientas de carga, como **Apache JMeter**, para medir el rendimiento de la aplicación y asegurarse de que cumpla con los estándares establecidos.

4. **Pruebas de Usabilidad**:

   - Se llevarán a cabo pruebas de usabilidad con usuarios finales para evaluar la experiencia general al interactuar con **MusicNet**. Esto incluirá la observación de cómo los usuarios navegan por la plataforma, así como la recopilación de retroalimentación sobre la interfaz y la facilidad de uso.
   - Las sesiones de pruebas de usabilidad se programarán antes del lanzamiento, y se utilizará la retroalimentación para realizar ajustes necesarios en la interfaz y la experiencia del usuario.

Proceso de Control de Calidad:

1. **Plan de Pruebas**: Se desarrollará un plan de pruebas que detalle todas las pruebas que se realizarán, incluyendo pruebas unitarias, de integración, de rendimiento y de usabilidad. Este plan incluirá cronogramas, recursos necesarios y criterios de aceptación.
2. **Ejecución de Pruebas**: Las pruebas se ejecutarán según el plan establecido, y se registrarán todos los resultados en un sistema de seguimiento de errores (como **Jira**) para gestionar y resolver los problemas encontrados.
3. **Revisión de Resultados**: Después de cada fase de pruebas, se llevará a cabo una reunión de revisión con el equipo para discutir los resultados, identificar errores y definir acciones correctivas. Se realizarán ajustes en el desarrollo según las lecciones aprendidas de las pruebas.
4. **Documentación de Calidad**: Todos los procesos de control de calidad, resultados de pruebas y acciones correctivas se documentarán de manera sistemática. Esta documentación se utilizará como referencia para el equipo y para futuros proyectos, asegurando que se mantengan altos estándares de calidad en el desarrollo.

Referencias

1. AubioJS Documentation. AubioJS Library for Audio Processing.
2. IEEE Standards Association (2008). ISO/IEC 12207:2008, Systems and Software Engineering - Software Life Cycle Processes.
3. IEEE Standards Association. (2024). *Music Packet Loss Concealment Challenge Documentation*.
4. Zhang, C., & Wu, Z. (2015). Adaptive Streaming of Video over Wireless Networks: A Survey. IEEE Transactions on Multimedia.
5. Google Design. Material Design Guidelines. Recuperado de [https://material.io/design]
6. Git Documentation. (2023). *Git - the simple guide.* https://git-scm.com/
7. Scrum Alliance (2023). Guía oficial de Scrum. Recuperado de [https://www.scrumalliance.org].
8. Kanban University (2023). Fundamentos de Kanban. Recuperado de https://www.kanbanuniversity.com.
9. Müller, R., et al. (2019). The Use of AI in Real-Time Applications. Journal of Applied AI.
10. International Organization for Standardization. (2019). ISO/IEC 40180:2019 - Information technology — Quality for educational software systems — Virtual Learning Environment (VLE) quality model. Geneva, Switzerland: ISO.
11. Schwaber, K., & Sutherland, J. (2017). The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game. Scrum.org. Disponible en

[https://www.scrumguides.org/]

12. Atlassian. (2023). Guía de Jira y Trello para el seguimiento de proyectos ágil. Atlassian. Disponible en: [https://www.atlassian.com]
13. Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business.* Blue Hole Press.
14. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Jeffries, R. (2001). *Manifesto for Agile Software Development.* Agile Alliance.
15. *Mozilla Developer Network (MDN).* (2023). *Web Audio API Documentation.* Mozilla. https://developer.mozilla.org/enUS/docs/Web/API/Web_Audio_API
16. Brossier, P. (2021). *Aubio, a library for audio and music analysis.* Aubio. https://aubio.org/
17. Thomas, C., & Stevenson, K. (2020). *Mastering Web Audio.* O'Reilly Media.
18. Rabiner, L. R., & Gold, B. (1975). *Theory and Application of Digital Signal Processing.* Prentice Hall.
19. International Telecommunication Union (ITU). (2019). *Recommendation ITU-T G.114: One-Way Transmission Time.*
20. Krug, S. (2014). Don't Make Me Think: A Common Sense Approach to Web Usability. New Riders.
21. Resig, J., & Bibeault, B. (2013). Secrets of the JavaScript Ninja. Manning Publications.
22. **Mackenzie, L., & Blackwell, R.** (2020). *Trello for Project Management: How to Organize and Manage Your Projects with Trello*.

23. **Van der Meulen, R.** (2018). *Google Sheets for Project Management: A Beginner's Guide*.
24. **Schwaber, K., & Sutherland, J.** (2020). *The Scrum Guide*. Scrum.org.
25. **Jørgensen, M.** (2014). *A review of studies on expert estimation of software development effort*. *Proceedings of the 2014 ACM conference on software engineering*.
26. **Levin, M., & Stepanek, T.** (2013). *Agile Estimating and Planning*. Prentice Hall.
27. **Cohn, M.** (2005). *Agile Estimating and Planning*. Prentice Hall.
28. Kerzner, H. (2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling (12th ed.). Wiley.
29. PMI (Project Management Institute). (2017). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (6th ed.). Project Management Institute.
30. Schwalbe, K. (2015). Information Technology Project Management. Cengage Learning.