Logotipo

Descripción generada automáticamente

DOCUMENTO DAS ARQUITECTURA DE SOFTWARE PROYECTO

Documento de Arquitectura de Software IEEE-1471-2000 Control del documento

• Nombre Proyecto: NeoCollab

Nombre Sub-Proyecto: NeoCollab

Arquitectura Proyecto Semestral

Integrantes: Diego San Martín

Nicolás Norambuena



Contenido

[**1.**](#_heading=h.gjdgxs) **Introducción** 3

[**2.**](#_heading=h.30j0zll) **Propósito** 3

[**3.**](#_heading=h.1fob9te) **Alcance** 3

[**4.**](#_heading=h.3znysh7) **Equipo de Trabajo-Actores del Desarrollo** 4

[**5.**](#_heading=h.2et92p0) **Recomendaciones de conformidad con esta práctica.** 4

[Para asegurar que la arquitectura del sistema cumpla con los requisitos y estándares establecidos, se recomiendan las siguientes prácticas: 4](#_heading=h.tyjcwt)

[1. Implementar revisiones de arquitectura cada xxxx para asegurar que se ajusten a los cambios de requisitos. 4](#_heading=h.3dy6vkm)

[2. Utilizar patrones de diseño desacoplados como MVC para modularidad y escalabilidad o por capas. 4](#_heading=h.1t3h5sf)

[3. Adoptar metodologías ágiles como Scrum para la entrega continua de mejoras y funcionalidades. 4](#_heading=h.4d34og8)

[**6.**](#_heading=h.2s8eyo1) **Referencias y estándares aplicables a este documento:** 5

[**7.**](#_heading=h.17dp8vu) **Definiciones, acrónimos y abreviaciones.** 5

[**8.**](#_heading=h.3rdcrjn) **DESCRIPCIÓN DE ARQUITECTURA:** La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. 6

[8.1 VISTAS: 6](#_heading=h.26in1rg)

[8.2 TIPOS DE VISTAS: 6](#_heading=h.lnxbz9)

[8.3 Framework Conceptual 7](#_heading=h.35nkun2)

[8.3.1 Descripción de la arquitectura en contexto 7](#_heading=h.1ksv4uv)

[8.3.1 Actores Usuarios y sus roles: 8](#_heading=h.44sinio)

[8.3.2 Actividades de arquitectura en el ciclo de vida 9](#_heading=h.2jxsxqh)

[8.3.3 Descripciones prácticas de arquitectura. 9](#_heading=h.z337ya)

[8.4 Documentación de la arquitectura 10](#_heading=h.3j2qqm3)

[8.5 Selección de puntos de vista de la arquitectura 10](#_heading=h.1y810tw)

[8.6 Vistas de la arquitectura 11](#_heading=h.4i7ojhp)

[UML-VISTA DE ESCENARIO 11](#_heading=h.2xcytpi)

[UML-VISTA LÓGICA-DIAGRAMA DE CLASES 12](#_heading=h.1ci93xb)

[UML-VISTA LÓGICA-DIAGRAMA DE BASE DE DATOS 14](#_heading=h.qsh70q)

[UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD 16](#_heading=h.1pxezwc)

[UML-VISTA DE DESPLIEGUE 18](#_heading=h.49x2ik5)

[UML-VISTA FÍSICA – DIAGRAMA INFRAESTRUCTURA 19](#_heading=h.2p2csry)

[Usabilidad y Calidad de software- Front-end -Vistas Principales 21](#_heading=h.147n2zr)

[Usabilidad y Calidad de software – Vistas específicas de usuario 22](#_heading=h.3o7alnk)

[8.7 Consistencia en la cantidad de vistas de la arquitectura. 23](#_heading=h.23ckvvd)

[8.7.1 DESCRIPCION DE MODULOS 23](#_heading=h.ihv636)

[8.7.2 DESCRIPCIÓN DE COMPONENTES 23](#_heading=h.32hioqz)

[8.7.3 DESCRIPCIÓN DE CONECTORES 23](#_heading=h.1hmsyys)

[9.1 Acceso a fuentes de información del proyecto 25](#_heading=h.41mghml)

# **Introducción**

El presente informe tiene como objetivo describir en detalle la arquitectura de un sistema llamado NeoCollab, diseñado para automatizar la administración de tareas y procesos comunes dentro de la colaboración musical en línea.

El sistema propuesto busca mejorar la interacción entre músicos y productores emergentes, facilitando la creación y el intercambio de pistas musicales a través de una plataforma móvil.

El sistema desarrollado será implementado utilizando Angular e Ionic para el frontend, con Firebase gestionando la autenticación, almacenamiento y base de datos. El sistema también proporcionará una interfaz intuitiva y accesible para todos los usuarios, asegurando que las colaboraciones se realicen de manera eficiente y organizada, basada en los estándares modernos de desarrollo de aplicaciones móviles.

# **Propósito**

El propósito de este documento es servir como guía para el diseño e implementación de la arquitectura del sistema NeoCollab. En él se detallan las decisiones técnicas, los componentes principales y los puntos de integración que permitirán implementar una solución robusta y escalable, capaz de gestionar múltiples comunidades bajo un esquema de administración unificado.

Este informe es esencial para los desarrolladores y stakeholders del proyecto, ya que les proporcionará una visión detallada de la estructura y comportamiento del sistema, así como las relaciones entre las diferentes funcionalidades y componentes. Asimismo, establece las bases para futuras expansiones y mejoras del sistema, considerando factores de disponibilidad, seguridad y usabilidad.

# **Alcance**

El sistema gestionará las siguientes áreas:

* Autenticación de usuarios: Implementación de un sistema seguro para el registro e inicio de sesión de los usuarios.
* Subida de beats: Funcionalidad que permitirá a los usuarios subir y gestionar sus beats en la nube.
* Grabación de audio: Posibilidad de grabar pistas de audio en tiempo real sobre los beats cargados por los usuarios.
* Muro social: Creación de un espacio donde los usuarios puedan publicar, comentar y compartir sus creaciones musicales.
* Interacción social: Implementación de funcionalidades como comentarios, likes y notificaciones en tiempo real para fomentar la interacción entre usuarios.
* Documentación y soporte: Provisión de documentación técnica y guías de usuario para facilitar el uso del sistema.

# **Equipo de Trabajo-Actores del Desarrollo**

El desarrollo del sistema requiere la participación de un equipo multidisciplinario con roles claramente definidos para asegurar que el proyecto se lleve a cabo de manera eficiente. A continuación, se presenta una tabla con los principales roles y sus responsabilidades:

| Rol | Responsabilidades |
| --- | --- |
| Cliente-Stakeholder | Proporcionar requisitos del negocio, dar feedback sobre el producto y validar las entregas del equipo de desarrollo. |
| Product Owner | Priorizar el backlog del producto, asegurar que el equipo entregue valor a los usuarios y ser el enlace entre los stakeholders y el equipo de desarrollo. |
| Desarrolladores Backend | Desarrollar la lógica del servidor, integrar servicios y APIs, y gestionar la autenticación y almacenamiento de datos. |
| Desarrolladores Frontend | Diseñar y desarrollar la interfaz de usuario utilizando Angular e Ionic, y asegurar la experiencia del usuario en la aplicación móvil. |
| DBA-Especialista en Base de Datos | Diseñar y gestionar la estructura de la base de datos en Firebase Firestore, asegurando la integridad y disponibilidad de los datos. |
| Tester/QA | Realizar pruebas funcionales y de usabilidad, identificar errores y asegurar que el sistema cumpla con los estándares de calidad establecidos. |
| Analista de Requisitos | Recopilar y analizar requisitos de los usuarios, documentar especificaciones y asegurar que se comprendan las necesidades del cliente. |
| Administrador Plataformas | Gestionar la infraestructura en la nube, asegurar el rendimiento del sistema y supervisar la implementación y mantenimiento de Firebase. |

# **Recomendaciones de conformidad con esta práctica.**

# Para asegurar que la arquitectura del sistema cumpla con los requisitos y estándares establecidos, se recomiendan las siguientes prácticas:

# Implementar revisiones de arquitectura cada dos semanas para asegurar que se ajusten a los cambios de requisitos.

# Utilizar patrones de diseño desacoplados como MVC para modularidad y escalabilidad o por capas.

# Adoptar metodologías ágiles como Scrum para la entrega continua de mejoras y funcionalidades.

# **Referencias y estándares aplicables a este documento:**

El documento sigue los siguientes estándares de calidad para asegurar la robustez y escalabilidad del sistema:

* **IEEE 830-1998 ST**: Estandarización de requisitos de software.
* **ISO 9126-2001**: Métricas de calidad del software.
* **Modelo 4+1 de Kruchten**: Modelo estructural basado en múltiples vistas de escenarios.
* **ATAM (Architecture Tradeoff Analysis Method)**: Metodología para evaluar y documentar decisiones arquitectónicas

# **Definiciones, acrónimos y abreviaciones.**

| Término | Definición |
| --- | --- |
| DAS | Documento de Arquitectura de Software |
| MVC | Modelo-Vista-Controlador, patrón de diseño utilizado para separar la lógica de negocio de la interfaz de usuario |
| UML | Unified Modeling Language, lenguaje gráfico para representar la arquitectura del sistema |
| CRUD | Operaciones básicas de Create, Read, Update, Delete para la gestión de datos. |

# **DESCRIPCIÓN DE ARQUITECTURA:** La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas. La arquitectura del sistema se presenta mediante el uso de vistas basadas en el Modelo 4+1 de Kruchten, con el fin de proporcionar una visión completa y estructurada del sistema desde diferentes perspectivas.

## VISTAS:

Cada vista se representará en un diagrama UML para describir la estructura y comportamiento del sistema.

### TIPOS DE VISTAS:

1. **Vista de Escenarios (Diagrama de Caso de Uso)**: Describe las funcionalidades del sistema y cómo interactúan los diferentes actores con él.
2. **Vista Lógica (Diagrama de Clases)**: Detalla la estructura de datos y las relaciones entre las principales entidades del sistema.
3. **Vista de Procesos (Diagrama de Actividad)**: Muestra el flujo de actividades dentro del sistema y cómo se coordinan los diferentes procesos.
4. **Vista de Despliegue (Diagrama de Componentes)**: Representa la distribución de componentes en la infraestructura física.
5. **Vista Física (Diagrama Topológico)**: Representa la infraestructura de red y la comunicación entre los diferentes servicios del sistema

## Framework Conceptual

### Descripción de la arquitectura en contexto

El sistema será diseñado y desarrollado utilizando un enfoque ágil con metodologías como Scrum, lo que permitirá la entrega incremental de funcionalidades, la retroalimentación constante y la capacidad de adaptarse a cambios en los requisitos a lo largo del ciclo de vida del proyecto. Esto asegurará que se puedan abordar necesidades emergentes y maximizar el valor del producto para los usuarios.

El diseño del sistema se basará en la separación de vistas, que ofrecerán diferentes perspectivas de la arquitectura para una comprensión integral. A continuación, se describen las principales vistas:

* **Vista de Escenarios**: Representada mediante diagramas de Casos de Uso, que describen las interacciones del usuario con el sistema y los flujos principales.
* **Vista Lógica**: Incluye el Modelo de Clases, que muestra las entidades y relaciones del sistema, así como el Modelo de Datos que detalla la estructura de la base de datos.
* **Vista de Procesos**: Utiliza Diagramas de Actividad para ilustrar los flujos internos del sistema y las interacciones entre los distintos módulos.
* **Vista de Despliegue**: Se implementará a través de Diagramas de Componentes o Modelos de Capas para reflejar la distribución de la lógica de negocio, la interfaz de usuario y la capa de acceso a datos.
* **Vista Física**: Utiliza Diagramas Topológicos o de Servicios para mostrar la distribución del sistema en el entorno físico, como servidores, redes y otros componentes de infraestructura.

Cada una de estas vistas está interconectada para garantizar la coherencia y la alineación con los requisitos de negocio. No hay ninguna vista separada de una implementación específica, sino que todas contribuyen a una visión holística de la arquitectura del sistema descrita en este documento.

Estas vistas se han creado utilizando el Lenguaje de Modelado Unificado (UML) en su versión 2.0 y han sido desarrolladas empleando herramientas de modelado estándar como **StarUML** y Lucidchart y en front-end **Canva, Miro, Figma u otro** para facilitar la creación de diagramas visuales que respalden la documentación arquitectónica.

El estilo arquitectónico se centra en una combinación de patrones de diseño orientados a servicios como SOA (Arquitectura orientada a servicios) y arquitectura por capas, en donde cada módulo del sistema se corresponde con un conjunto específico de casos de uso que resuelven los procesos definidos para la operación.

Estos procesos corresponden a:

* **Gestión de beats**: Permite a los usuarios subir, gestionar y organizar sus beats en la plataforma, asegurando un acceso fácil y eficiente a sus creaciones.
* **Grabación de pistas**: Facilita a los usuarios la grabación de audio en tiempo real sobre los beats, gestionando la sincronización y el almacenamiento de las pistas grabadas.
* **Interacción social**: Gestión de interacciones entre usuarios a través de comentarios, likes y notificaciones, promoviendo la colaboración y el feedback en el muro social.
* **Muro social**: Administración de publicaciones de usuarios, permitiendo compartir colaboraciones musicales, comentarios y reacciones.
* **Seguridad y autenticación**: Gestión de la autenticación de usuarios y protección de datos, asegurando que las interacciones en la plataforma sean seguras y confiables.

### Actores Usuarios y sus roles:

Este documento representa la identificación de Actores/Usuarios Stakeholders y sus roles a partir de la interpretación de los casos de uso del Negocio asociados.

|  |  |  |
| --- | --- | --- |
| ACTOR | ROL / CARGO | FUNCION |
| Diego San Martín | Product Owner | Prioriza las necesidades del cliente y asegura la entrega de valor a los usuarios. |
| Nicolás Norambuena | Scrum Master | Facilita las prácticas ágiles, gestiona impedimentos y coordina el equipo de desarrollo. |
| Diego San Martín / Nicolás Norambuena | Backend & Frontend Developer | Desarrolla e integra el backend y frontend de la aplicación, implementando funcionalidades clave. |
| Músicos y Productores Emergentes | Usuarios Finales | |  | | --- | |  |  |  | | --- | | Utilizan la plataforma para colaborar, compartir su música y recibir retroalimentación. | |
| Diego San Martín / Nicolás Norambuena | Tester | Realiza pruebas de funcionalidad y usabilidad para asegurar la calidad del producto. |

### 

### 8.3.2 Actividades de arquitectura en el ciclo de vida

El desarrollo de la arquitectura del sistema NeoCollab se abordó utilizando un ciclo de vida **iterativo incremental** basado en **Scrum**. Esta metodología fue seleccionada porque permite realizar entregas parciales y frecuentes de software funcional, facilitando la validación temprana de los requisitos por parte de los stakeholders y el product owner con la adaptación a cambios en el entorno de negocio.

El ciclo de vida se organizó en **sprints** de 2 a 3 semanas, con cada iteración centrada en la construcción y mejora de un módulo específico del sistema (por ejemplo, el módulo de gestión de gastos comunes o el módulo de reservas). Al final de cada sprint, se realiza una revisión detallada de los resultados para asegurar la alineación con las expectativas del cliente y se priorizan las tareas para el siguiente sprint.

### 8.3.3 Descripciones prácticas de arquitectura.

Durante el proceso de desarrollo, la arquitectura del sistema fue escalada desde un **escenario principal de caso de uso** hasta un diseño modular a través de los siguientes pasos:

1. Análisis de Requerimientos: Se analizaron los requisitos iniciales del sistema para identificar los casos de uso principales, tales como "Registrar Usuarios", "Subir Beats", "Grabar Pistas de Audio" y "Interactuar en el Muro Social".
2. Definición de la Vista Lógica: A partir del escenario principal, se identificaron las entidades y relaciones claves en un diagrama de clases UML, representando el núcleo del sistema.
3. Modelo de Datos: Se desarrolló un diagrama de base de datos para definir la estructura de la información, manteniendo la coherencia con la vista lógica.
4. Vista de Procesos: Se modelaron los flujos de las actividades y transacciones principales del sistema mediante diagramas de actividad.
5. Despliegue: Se definió la infraestructura física y lógica para alojar el sistema en la nube, detallando los servicios necesarios, como balanceadores de carga, servidores de aplicaciones y bases de datos distribuidas.
6. Definición de Planes de pruebas y técnicas de pruebas testing.

## Documentación de la arquitectura

* **ISO/IEC 25010**: Se utilizó para evaluar la calidad del sistema en términos de funcionalidad, seguridad y eficiencia.
* **IEEE 1471-2000**: Proporcionó la guía para la representación de vistas arquitectónicas y la identificación de stakeholders.
* **Modelo 4+1 de Kruchten**: Estructura la arquitectura en vistas lógicas, de desarrollo, de procesos y física, facilitando la visualización desde múltiples perspectivas.
* **Planes de prueba**: (Declarar los tipos de pruebas a realizar)

## Selección de puntos de vista de la arquitectura

|  |  |  |
| --- | --- | --- |
| VISTAS | UML | Cantidad |
| Escenario Principal | Diagrama de Caso de uso | 1 (2-3 los necesarios) |
| Vista Lógica | Diagrama de Clases  MER | 1  1 |
| Vista de Proceso | Diagrama de Actividad:  Nombre 1.  Nombre 2  Nombre 3. | 1 (2-3, los necesarios) |
| Vista Física | Diagrama Topológico de Servicios | 1 |
| Vista de Despliegue | Diagrama de componentes  Diagrama de Capas (MVC) | 1  1 |

## Vistas de la arquitectura

Ejemplos:

|  |
| --- |
| UML-VISTA DE ESCENARIO |
| Diagrama de Caso de uso extendido /Subproyecto [Inserte nombre del Subproyecto asignado] |
|  |

**Gestión de Usuarios:** NeoCollab permite que los usuarios no registrados se registren para acceder a las funcionalidades de la plataforma. Los usuarios registrados pueden iniciar sesión, recuperar su contraseña y acceder a diversas características según su rol (músico, productor o administrador).

**Publicación en el Muro Social:** Los usuarios registrados pueden publicar en el muro social, lo cual permite compartir sus ideas, proyectos y avances musicales con la comunidad. Esta funcionalidad también incluye la opción de comentar publicaciones de otros usuarios y dar "Me gusta" a las publicaciones que deseen apoyar.

**Recomendaciones Personalizadas:** NeoCollab ofrece recomendaciones personalizadas para los usuarios, basándose en sus preferencias y actividad en la plataforma. Esta funcionalidad incluye notificaciones automáticas sobre recomendaciones de pistas o beats, generando un flujo constante de inspiración.

**Gestión de Beats:** Los productores musicales pueden subir beats a la plataforma, recibiendo una confirmación de subida exitosa. También tienen la posibilidad de gestionar sus beats ya subidos, incluyendo la edición o eliminación de los mismos cuando lo consideren necesario.

**Grabación y Ajuste de Audio:** Los músicos pueden utilizar NeoCollab para grabar audio directamente en la plataforma, con la opción de previsualizar sus grabaciones antes de guardarlas. Además, se les permite ajustar aspectos específicos de la grabación para lograr un sonido óptimo y personalizar su contenido antes de compartirlo.

**Moderación y Monitoreo del Sistema:** El administrador del sistema tiene el rol de monitorear las actividades dentro de NeoCollab y moderar el contenido para asegurar un ambiente seguro y respetuoso. Esto incluye la posibilidad de eliminar contenido inapropiado y gestionar las interacciones en el muro social.

**Alertas y Notificaciones Automáticas:** NeoCollab envía notificaciones automáticas a los usuarios para mantenerlos al tanto de comentarios, "Me gusta" y otras interacciones relevantes en sus publicaciones. Además, el sistema notifica a los usuarios sobre eventos importantes, como recomendaciones y mensajes destacados en el muro.

|  |
| --- |
| UML-VISTA LÓGICA-DIAGRAMA DE CLASES |
| [Diagrama de Clases] |
| Interfaz de usuario gráfica  Descripción generada automáticamente |

El Diagrama de Clases para la plataforma NeoCollab muestra las clases esenciales, sus atributos, métodos y las relaciones entre ellas. Este diagrama ayuda a modelar la estructura y el comportamiento del sistema de manera orientada a objetos, facilitando la interacción y la colaboración musical entre usuarios.

Elementos Principales

* Clases:
  + Usuario:
    - Atributos: ID\_Usuario, Nombre, Correo Electrónico, Contraseña, Fecha de Registro, Tipo de Usuario.
    - Métodos: iniciarSesion(), registrarse(), recuperarContraseña().
  + Beat:
    - Atributos: ID\_Beat, Título, Archivo de Beat, Género Musical, Fecha de Subida, Portada del Beat.
    - Métodos: subirBeat(), editarBeat(), eliminarBeat().
  + Pista:
    - Atributos: ID\_Pista, Título, Archivo de la Pista, Fecha de Grabación.
    - Métodos: grabarPista(), previsualizarPista(), eliminarPista().
  + Publicación en el Muro:
    - Atributos: ID\_Publicación, Contenido, Fecha de Publicación.
    - Métodos: publicar(), editarPublicación(), eliminarPublicación().
  + Comentario:
    - Atributos: ID\_Comentario, Contenido, Fecha de Publicación.
    - Métodos: comentar(), editarComentario(), eliminarComentario().
  + Like:
    - Atributos: ID\_Like, Fecha.
    - Métodos: darLike(), quitarLike().
  + Perfil de Usuario:
    - Atributos: ID\_Perfil, Descripción, Foto de Perfil, Redes Sociales.
    - Métodos: actualizarPerfil(), verPerfil().
  + Notificación:
    - Atributos: ID\_Notificación, Mensaje, Tipo de Notificación, Fecha de Emisión.
    - Métodos: enviarNotificación(), verNotificación(), eliminarNotificación().

Relaciones:

* **Usuario** tiene una relación de asociación con **Perfil de Usuario**, permitiendo que cada usuario mantenga su perfil personalizado.
* **Usuario** puede crear múltiples **Beats y Pistas**, que se gestionan individualmente en la plataforma.
* **Publicación** **en el Muro** permite que los usuarios publiquen contenido, que puede incluir comentarios y recibir likes, creando un entorno interactivo.
* **Comentario** está asociado a **Publicación en el Muro** y permite la interacción directa con cada publicación.
* **Notificación** se genera para **Usuario** en función de interacciones como comentarios, likes y recomendaciones en el muro social, manteniendo a los usuarios informados de la actividad en su contenido.

|  |
| --- |
| UML-VISTA LÓGICA-DIAGRAMA DE BASE DE DATOS |
| [Diagrama de BD - MER] |
| Interfaz de usuario gráfica  Descripción generada automáticamente |

El Modelo Entidad-Relación (MER) de NeoCollab

El Modelo Entidad-Relación (MER) para la plataforma NeoCollab define las entidades esenciales, sus atributos y las relaciones que existen entre ellas. Este modelo organiza la información clave para gestionar la red social y las interacciones musicales en la plataforma, estructurando los datos de usuarios, publicaciones, beats, notificaciones, y comentarios.

Elementos Principales

* Entidades:
  + Usuario:
    - Atributos: ID\_Usuario (PK), Nombre, Correo Electrónico, Contraseña, Fecha de Registro, Tipo de Usuario (REG/ADM).
  + Beat:
    - Atributos: ID\_Beat (PK), Título, Archivo de Beat, Género Musical, Fecha de Subida, Portada del Beat, ID\_Usuario (FK).
  + Pista:
    - Atributos: ID\_Pista (PK), Título, Archivo de la Pista, ID\_Beat (FK1), ID\_Usuario (FK2), Fecha de Grabación.
  + Publicación en el Muro:
    - Atributos: ID\_Publicación (PK), Contenido, Fecha de Publicación, ID\_Usuario (FK), ID\_Beat (FK).
  + Comentario:
    - Atributos: ID\_Comentario (PK), Contenido, Fecha de Publicación, ID\_Usuario (FK), ID\_Publicación (FK).
  + Like:
    - Atributos: ID\_Like (PK), ID\_Usuario (FK), ID\_Publicación (FK).
  + Perfil de Usuario:
    - Atributos: ID\_Perfil (PK), Descripción, Foto de Perfil, Redes Sociales, ID\_Usuario (FK).
  + Notificación:
    - Atributos: ID\_Notificación (PK), Mensaje de Notificación, Tipo de Notificación, Fecha de Emisión, ID\_Usuario (FK).

Relaciones:

* **Usuario** puede tener un perfil de usuario detallado que incluye su descripción y redes sociales.
* **Usuario** puede crear y gestionar múltiples **Beats** que luego puede compartir en su perfil.
* **Beat** puede asociarse a una **Pista**, que también está vinculada al usuario que la grabó.
* **Publicación** en el Muro permite a los usuarios publicar contenido que puede incluir un beat específico y recibir comentarios, likes y recomendaciones.
* **Comentario** está asociado a una publicación y es generado por un usuario.
* **Like** permite a los usuarios interactuar con publicaciones en el muro social.
* **Notificación** informa a los usuarios sobre interacciones en sus publicaciones, como nuevos comentarios, likes o recomendaciones.

|  |
| --- |
| UML-VISTA DE PROCESO-DIAGRAMAS DE ACTIVIDAD |
| [Diagramas de Actividad] |
| Texto, Carta  Descripción generada automáticamente |

El diagrama de actividad representa el flujo de actividades dentro de NeoCollab, una aplicación de colaboración musical para músicos y productores. Este diagrama es útil para visualizar los pasos secuenciales y las acciones que los usuarios pueden realizar en la plataforma, desde la creación de una cuenta hasta la interacción con otros usuarios y la gestión de contenidos.

**Componentes:**

1. **Inicio:** El usuario completa el formulario de registro en NeoCollab.
2. **Validación y Almacenamiento de Usuario:** El sistema valida los datos del usuario y almacena la información en su base de datos.
3. **Iniciar Sesión:** El usuario ingresa su correo y contraseña para acceder a la plataforma.
4. **Verificación de Credenciales:** El sistema verifica las credenciales y, si son correctas, permite el acceso al usuario.
5. **Publicación en el Muro:** El usuario puede comentar, dar "Me gusta" o compartir publicaciones en el muro social.
   * **Comentario o "Me gusta":** El sistema registra la interacción y envía una notificación a los seguidores del usuario sobre la actividad reciente.
6. **Grabación de Audio:** El usuario selecciona la opción de grabación para crear un nuevo contenido musical.
   * **Ajuste y Edición de Grabación:** El usuario tiene la opción de previsualizar y ajustar la grabación antes de guardarla.
7. **Almacenamiento de Grabación:** El sistema almacena la grabación y la vincula al perfil del usuario, haciéndola visible para otros usuarios en su red.
8. **Moderación y Revisión de Contenidos:** El administrador revisa las actividades dentro de la plataforma para asegurar el cumplimiento de las normas.
   * **Edición o Eliminación de Contenido:** Si se detecta contenido inapropiado, el administrador puede editar o eliminar la publicación para mantener el ambiente adecuado en la comunidad.
9. **Fin:** El flujo concluye con la grabación almacenada en el perfil del usuario y la moderación activa del administrador, lo que contribuye a una experiencia segura y organizada para todos los miembros de NeoCollab.

|  |
| --- |
| UML-VISTA DE DESPLIEGUE |
| Sistema de NeoCollab |
| Imagen que contiene Diagrama |

**Servidor de Aplicaciones NeoCollab:**

* **App web NeoCollab:** Proporciona una interfaz intuitiva para los usuarios del sistema, incluyendo músicos y administradores. Permite funcionalidades como la subida y gestión de beats, publicación en el muro social, interacción entre usuarios (comentarios y "me gusta"), y ajustes de perfil. Este servidor es el núcleo de la aplicación, gestionando las acciones de los usuarios y coordinando la comunicación con la base de datos y el sistema de notificaciones.

**Servidor de base de datos:**

* **BD NeoCollab (Firebase):** Almacena todos los datos esenciales de la aplicación, como información de usuarios, beats, publicaciones, comentarios y registros de actividad en el sistema. Este servidor es crucial para el funcionamiento de NeoCollab, permitiendo la recuperación y almacenamiento de datos en tiempo real para mantener la experiencia de usuario fluida y actualizada.

**Servidor de notificaciones:**

* **Sistema de notificaciones:** Responsable de enviar notificaciones automáticas a los usuarios, alertando sobre nuevas interacciones en sus publicaciones, recordatorios de eventos, y actualizaciones importantes. Este servidor permite que los usuarios estén informados en tiempo real, mejorando la interactividad y el compromiso dentro de la comunidad de NeoCollab.

**Usuarios:**

* **Interfaz de usuario registrado:** Ofrece a los músicos y productores un espacio para interactuar con la plataforma, subir beats, comentar en publicaciones, y personalizar su perfil.
* **Interfaz de administrador:** Brinda a los administradores herramientas para moderar el contenido, gestionar usuarios y monitorear actividades dentro de la plataforma.

|  |
| --- |
| UML-VISTA FÍSICA – DIAGRAMA INFRAESTRUCTURA |
| Sistema Gestión De Condominios |
| Diagrama  Descripción generada automáticamente |

**Centro de datos de NeoCollab**: Es el núcleo del sistema donde se aloja la infraestructura principal de NeoCollab. Incluye servidores dedicados para la aplicación web y el almacenamiento de datos en Firebase, así como un servidor de notificaciones para mantener a los usuarios informados en tiempo real. Todos estos elementos están organizados en una estructura de redes para garantizar una comunicación eficiente y segura entre los diferentes servicios.

**Red de aplicaciones NeoCollab**: Contiene el **servidor de aplicaciones NeoCollab**, que hospeda la lógica y la funcionalidad principal de la aplicación web, y la **base de datos NeoCollab (Firebase)**, que almacena toda la información relevante del usuario y sus interacciones. Estos componentes permiten a la aplicación manejar eficientemente el flujo de datos y asegurar la disponibilidad de información para todos los usuarios.

**Red de usuarios y clientes**: Esta red está conformada por el **servidor de notificaciones**, que es responsable de enviar notificaciones en tiempo real a los usuarios, como actualizaciones de publicaciones, interacciones en el muro social y mensajes directos. Esta red permite una comunicación fluida y oportuna entre la aplicación y los usuarios.

**Red del usuario final**: Incluye las interfaces de usuario necesarias para la interacción con el sistema, específicamente la **interfaz de usuario registrado** y la **interfaz de administrador**. Estas interfaces permiten a los usuarios interactuar con NeoCollab, ya sea para subir y gestionar contenido, o para moderar y supervisar la actividad en la plataforma, en el caso de los administradores.

**Conexiones externas**: Aunque no se muestra explícitamente, el acceso a NeoCollab se realiza a través de Internet, permitiendo que los usuarios se conecten desde sus dispositivos personales y accedan a la aplicación web alojada en el **servidor de aplicaciones NeoCollab**. Esta infraestructura asegura que los datos se mantengan seguros y que las notificaciones se entreguen de manera confiable a todos los usuarios.

|  |
| --- |
| Usabilidad y Calidad de software- Front-end -Vistas Principales |
| Vistas principales del sistema – Vista Home Principal de entrada |
|  |

La Vista Front-End de NeoCollab está diseñada para ofrecer una experiencia intuitiva y atractiva, ideal para una comunidad musical. Con una paleta de colores oscuros y morados, el diseño es moderno y profesional, permitiendo a los usuarios enfocarse en el contenido. La interfaz organiza las funciones clave —como subir beats, grabar y socializar— en menús y botones accesibles, facilitando una navegación fluida y eficiente.

Se han integrado iconos intuitivos para identificar rápidamente las funciones principales, mejorando la usabilidad. Además, la aplicación es completamente adaptable a dispositivos móviles, asegurando una experiencia uniforme y accesible en cualquier plataforma.

|  |
| --- |
| Usabilidad y Calidad de software – Vistas específicas de usuario |
| Vistas principales del sistema – Vista Principal de Usuario a escritorio de trabajo principal de su componente de sistema |
|  |

Cada tipo de usuario en NeoCollab cuenta con una interfaz personalizada que permite acceso a las funcionalidades específicas de su rol:

* **Usuario registrado:** Puede subir beats, grabar audio, interactuar en el muro social, recibir recomendaciones y gestionar su perfil.
* **Productor musical:** Acceso adicional para gestionar y editar beats, con opciones avanzadas para subir contenido musical.
* **Administrador:** Control completo del sistema, incluyendo la supervisión de actividad en la plataforma y la moderación de contenido.

## Consistencia en la cantidad de vistas de la arquitectura.

Cada vista de la arquitectura de NeoCollab está diseñada para alinearse con los requisitos funcionales y no funcionales, asegurando coherencia y evitando redundancias en toda la estructura del sistema.

### DESCRIPCION DE MODULOS

Ejemplos:

|  |  |  |
| --- | --- | --- |
| Nombre del módulo | Descripción | Componentes incluidos |
| Gestión de Gasto Comunes | Permite calcular y registrar gastos comunes. | Módulo de cálculo, Módulo de reportes, Módulo de facturación |
| Reserva de Espacios Comunes | Controla la reserva y uso de espacios comunes | Módulo de reservas, Módulo de notificaciones, Módulo de reportes |
| Gestión de Usuarios | Administración de cuentas y roles | Módulo de registro, Módulo de autenticación, Módulo de perfiles |

### DESCRIPCIÓN DE COMPONENTES

|  |  |  |
| --- | --- | --- |
| Nombre del Componente | Descripción | Componentes Relaciones |
| Calculadora de Gastos | Realizar cálculos de gastos basados en tarifas y consumos | Módulo de Gastos, módulo de Reportes |
| Motor de Notificaciones | Envía notificaciones por correo y mensajes emergentes | Módulo de Reservas, Módulo de Gastos |
| Gestor de Usuarios | Maneja la creación y eliminación de Usuarios | Módulo de Gestión de Usuarios, Módulo de Seguridad |

### DESCRIPCIÓN DE CONECTORES

|  |  |  |
| --- | --- | --- |
| Nombre del Conector | Tipo | Propósito |
| Rest API | Comunicación | Transferencia de datos entre módulos |
| Socket.io | Comunicación | Actualizaciones en tiempo real |

* + - 1. **Arquitectura lógica**
      2. Performances

El sistema está diseñado para manejar una alta concurrencia, soportando más de 1000 usuarios simultáneamente sin comprometer el rendimiento. Para lograr tiempos de respuesta inferiores a 2 segundos por transacción, se implementarán las siguientes estrategias:

* **Escalabilidad Horizontal**: Se utilizarán contenedores Docker para desplegar microservicios en múltiples instancias en la infraestructura de AWS, permitiendo la adición de recursos según la demanda.
* **Balanceo de Carga**: Se implementará un balanceador de carga que distribuye las solicitudes de los usuarios entre las diferentes instancias de los microservicios, asegurando una distribución equitativa del tráfico y evitando sobrecargas en un único servicio.
* **Optimización de Consultas**: Se aplicarán técnicas de optimización en las consultas a la base de datos PostgreSQL, incluyendo índices y consultas asíncronas para reducir los tiempos de respuesta.
  + - 1. Criterios de Calidad

El sistema debe cumplir con los siguientes criterios de calidad:

* **Disponibilidad 24/7**: La arquitectura se diseñará para asegurar un tiempo de actividad continuó, implementando redundancia y recuperación ante desastres en la infraestructura de AWS.
* **Respuesta Rápida**: Todas las transacciones deberán completarse en menos de 2 segundos, lo que se logrará mediante el uso de técnicas de caché y un diseño eficiente de la API.
* **Alta Seguridad**: Se implementará cifrado AES-256 para proteger datos sensibles, así como autenticación y autorización robustas para garantizar la integridad de la información y la privacidad del usuario.
  + - 1. Detalles de la implementación

La implementación del sistema se basará en una arquitectura de microservicios, utilizando las siguientes tecnologías:

* **Backend**: Node.js con Express para la creación de APIs RESTful.
* **Frontend**: React junto con Material-UI para la creación de una interfaz de usuario moderna y responsiva.
* **Base de Datos**: PostgreSQL para la gestión eficiente de datos estructurados.
* **Infraestructura**: AWS utilizando contenedores Docker para una gestión flexible y escalable de los microservicios.
  + - 1. Lenguajes y plataformas

Las tecnologías y versiones específicas que se utilizarán en el desarrollo del sistema incluyen:

* **Node.js**: v10.x
* **Express**: v4.x
* **React**: v16.x
* **Material-UI**: v4.x
* **PostgreSQL**: v9.6
* **Docker**: v17.x
* **AWS**: Servicios de EC2, RDS y ECS.
* **Python**: v3.7
* **Visual Studio Code**: v1.50

## Acceso a fuentes de información del proyecto

Para acceder a la carpeta de evidencias acumuladas y otros documentos relevantes del proyecto, puede utilizar el siguiente enlace:  
  
Insertar link: Carpeta de Evidencia del Proyecto  
  
Esta carpeta incluye documentación técnica, diagramas, pruebas de rendimiento y cualquier otra evidencia relacionada con el desarrollo y la implementación del sistema