



**UNIVERSIDAD DE
CÓRDOBA**



LICENCIATURA EN
INFORMÁTICA

Acreditada de Alta Calidad
MEN Res. 10710 25/05/17

Documento
técnico para
proyectos de
Diseño de
Software

Documento de Propuesta de Diseño de Software I, II y III

Creación traductor a lenguaje de señas

Autores:

Leonardo Triana Coronado / Ltrianacoronado18@correo.unicordoba.edu.co /

José Camilo Orozco Avilez / jorozcoavilez@correo.unicordoba.edu.co

Jesús David Vega Lomineth / jvegalomineth10@correo.unicordoba.edu.co /

Jorge Esteban Regino Montalvo / jreginomontalvo31@correo.unicordoba.edu.co /

Estivinson Buelvas Carrascal ebuelvascarrascal54@correo.unicordoba.edu.co /

Camilo Andrés Muñoz Arroyo Cmunozarrovo54@correo.unicordoba.edu.co /

Víctor León Patrón / vleonpatron06@correo.unicordoba.edu.co /

Link repositorio:



Reseña

El lenguaje de señas es una forma de comunicación fundamental que todos deberíamos comprender por ese motivo se pretende crear un software educativo que utiliza tecnología de reconocimiento de voz para traducir a lenguaje de señas representando por medio de imágenes. Este tipo de herramienta sería invaluable para personas sordas o con discapacidad auditiva, ya que les permitiría acceder al contenido auditivo de una manera adaptada a sus capacidades. El software emplea algoritmos de procesamiento de voz para reconocer y traducir el habla en tiempo real, mostrando las representaciones visuales del lenguaje de señas en la pantalla.

ETAPA 1 DISEÑO DE LA APLICACIÓN Y ANÁLISIS DE REQUISITOS	5
INTRODUCCIÓN	5
PROPÓSITO DEL DOCUMENTO	5
ALCANCE DEL PROYECTO	5
DEFINICIONES Y ACRÓNIMOS	5
DESCRIPCIÓN GENERAL	5
OBJETIVOS DEL SISTEMA	5
FUNCIONALIDAD GENERAL	5
USUARIOS DEL SISTEMA	5
RESTRICCIONES	5
REQUISITOS FUNCIONALES	5
CASOS DE USO	5
DESCRIPCIÓN DETALLADA DE CADA CASO DE USO	5
DIAGRAMAS DE FLUJO DE CASOS DE USO	5
PRIORIDAD DE REQUISITOS	5
REQUISITOS NO FUNCIONALES	5
REQUISITOS DE DESEMPEÑO	5
REQUISITOS DE SEGURIDAD	6
REQUISITOS DE USABILIDAD	6
REQUISITOS DE ESCALABILIDAD	6
MODELADO E/R	6
DIAGRAMA DE ENTIDAD-RELACIÓN	6
DESCRIPCIÓN DE ENTIDADES Y RELACIONES	6
REGLAS DE INTEGRIDAD	6
ANEXOS (SI ES NECESARIO)	6
DIAGRAMAS ADICIONALES	6
REFERENCIAS	6
ETAPA 2: PERSISTENCIA DE DATOS CON BACKEND	7
INTRODUCCIÓN	7
PROPÓSITO DE LA ETAPA	7
ALCANCE DE LA ETAPA	7

DEFINICIONES Y ACRÓNIMOS	7
DISEÑO DE LA ARQUITECTURA DE BACKEND	7
DESCRIPCIÓN DE LA ARQUITECTURA PROPUESTA	7
COMPONENTES DEL BACKEND	7
DIAGRAMAS DE ARQUITECTURA	7
ELECCIÓN DE LA BASE DE DATOS	7
EVALUACIÓN DE OPCIONES (SQL o NoSQL)	7
JUSTIFICACIÓN DE LA ELECCIÓN	7
DISEÑO DE ESQUEMA DE BASE DE DATOS	7
IMPLEMENTACIÓN DEL BACKEND	7
ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN	7
CREACIÓN DE LA LÓGICA DE NEGOCIO	7
DESARROLLO DE ENDPOINTS Y APIs	7
AUTENTICACIÓN Y AUTORIZACIÓN	7
CONEXIÓN A LA BASE DE DATOS	8
CONFIGURACIÓN DE LA CONEXIÓN	8
DESARROLLO DE OPERACIONES CRUD	8
MANEJO DE TRANSACCIONES	8
PRUEBAS DEL BACKEND	8
DISEÑO DE CASOS DE PRUEBA	8
EJECUCIÓN DE PRUEBAS UNITARIAS Y DE INTEGRACIÓN	8
MANEJO DE ERRORES Y EXCEPCIONES	8
ETAPA 3: CONSUMO DE DATOS Y DESARROLLO FRONTEND	9
INTRODUCCIÓN	9
PROPÓSITO DE LA ETAPA	9
ALCANCE DE LA ETAPA	9
DEFINICIONES Y ACRÓNIMOS	9
CREACIÓN DE LA INTERFAZ DE USUARIO (UI)	9
DISEÑO DE LA INTERFAZ DE USUARIO (UI) CON HTML Y CSS	9
CONSIDERACIONES DE USABILIDAD	9
MAQUETACIÓN RESPONSIVA	9

PROGRAMACIÓN FRONTEND CON JAVASCRIPT (JS)	9
DESARROLLO DE LA LÓGICA DEL FRONTEND	9
MANEJO DE EVENTOS Y COMPORTAMIENTOS DINÁMICOS	9
USO DE BIBLIOTECAS Y FRAMEWORKS (SI APLICABLE)	9
CONSUMO DE DATOS DESDE EL BACKEND	9
CONFIGURACIÓN DE CONEXIONES AL BACKEND	9
OBTENCIÓN Y PRESENTACIÓN DE DATOS	9
ACTUALIZACIÓN EN TIEMPO REAL (SI APLICABLE)	9
INTERACCIÓN USUARIO-INTERFAZ	10
MANEJO DE FORMULARIOS Y VALIDACIÓN DE DATOS	10
IMPLEMENTACIÓN DE FUNCIONALIDADES INTERACTIVAS	10
MEJORAS EN LA EXPERIENCIA DEL USUARIO	10
PRUEBAS Y DEPURACIÓN DEL FRONTEND	10
DISEÑO DE CASOS DE PRUEBA DE FRONTEND	10
PRUEBAS DE USABILIDAD	10
DEPURACIÓN DE ERRORES Y OPTIMIZACIÓN DEL CÓDIGO	10
IMPLEMENTACIÓN DE LA LÓGICA DE NEGOCIO EN EL FRONTEND	10
MIGRACIÓN DE LA LÓGICA DE NEGOCIO DESDE EL BACKEND (SI NECESARIO)	10
VALIDACIÓN DE DATOS Y REGLAS DE NEGOCIO EN EL FRONTEND	10
INTEGRACIÓN CON EL BACKEND	10
VERIFICACIÓN DE LA COMUNICACIÓN EFECTIVA CON EL BACKEND	10
PRUEBAS DE INTEGRACIÓN FRONTEND-BACKEND	10

Etapas 1 Diseño de la Aplicación y Análisis de Requisitos

Introducción

Propósito del Documento

El presente documento tiene como finalidad documentar el proceso de diseño, análisis e implementación de software de tipo educativo, comercial, OVA, componente o módulo de aplicaciones. Se divide en tres etapas para facilitar el entendimiento y aplicación a gran escala en la asignatura de diseño de software.

- Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

Esta etapa cumple la tarea de recoger todas las competencias desarrolladas en todas las áreas de formación del currículo de la licenciatura en Informática y Medios Audiovisuales y ponerlas a prueba en el diseño y análisis de un producto educativo que se base en las teorías de aprendizaje estudiadas, articule las estrategias de enseñanza con uso de TIC y genere innovaciones en educación con productos interactivos que revelen una verdadera naturaleza educativa. Estos productos deben aprovechar las fortalezas adquiridas en las áreas de tecnología e informática, técnicas y herramientas, medios audiovisuales y programación y sistemas, para generar productos software interactivos que permitan a los usuarios disfrutar de lo que aprenden, a su propio ritmo. Todo esto en el marco de un proceso metodológico (metodologías de desarrollo de software como MODESEC, SEMLI, etc.) que aproveche lo aprendido en la línea de gestión y lo enriquezca con elementos de la Ingeniería de Software.

- Etapa 2: Persistencia de Datos con Backend – Servidor

En la etapa 2 se continúa con los lineamientos de la etapa 1, para seguir adicionando elementos de diseño e implementación de software, enfocados en el desarrollo de APIs, servidores o microservicios que permitan soportar aplicaciones cliente del software educativo; en este sentido, el curso presenta los conceptos de los sistemas de bases de datos, su diseño lógico, la organización de los sistemas manejadores de bases de datos, los lenguaje de definición de datos y el lenguaje de manipulación de datos SQL y NoSQL; de tal manera que los estudiantes adquieran las competencias para analizar, diseñar y desarrollar aplicaciones para gestionar y almacenar grandes cantidades de datos, mediante el uso de técnicas adecuadas como el diseño y modelo lógico y físico de base datos, manejo de los sistemas de gestión de bases de datos, álgebra relacional, dominio del lenguaje SQL como herramienta de consulta, tecnología cliente / servidor; igualmente, se definirán los elementos necesarios para el acceso a dichas bases de datos, como la creación del servidor API, utilizando tecnologías de vanguardia como node.js, express, Nest.js, Spring entre otros; para, finalmente converger en el despliegue de la API utilizando servicios de hospedaje en la nube, preferiblemente gratuitos. También podrá implementar servidores o API 's con inteligencia artificial o en su defecto crear una nueva capa que consuma y transforme los datos obtenidos de la IA. El desarrollo del curso se trabajará por proyectos de trabajo colaborativo que serán evaluados de múltiples maneras, teniendo en cuenta más el proceso que el resultado.

Etapa 3: Consumo de Datos y Desarrollo Frontend – Cliente

La etapa 3 el estudiante está en capacidad de establecer la mejor elección de herramientas de consumo de datos y técnicas en aras de lograr el mejor producto a nivel de software o hardware acorde a los requerimientos funcionales y no funcionales del problema a solucionar. En este punto el estudiante puede consumir los datos a través de un cliente que puede ser una aplicación de celular, una aplicación de escritorio, una página web, IoT(internet de las cosas) o incluso, artefactos tecnológicos. El diseño gráfico es de los requisitos esenciales en la capa de presentación, por lo tanto, se requieren los cursos de diseño gráfico vistos previamente. Los elementos anteriores nos permiten elegir el paradigma y tecnología para desarrollar nuestras aplicaciones, teniendo en cuenta que podríamos desarrollar aplicaciones de tipo cliente

Alcance del Proyecto

El alcance del proyecto consiste en desarrollar una herramienta de software que traduce audios en tiempo real a lenguaje de señas que se representarán con imágenes animadas. Esta herramienta educativa y de accesibilidad para todos, será invaluable para personas sordas o con discapacidad auditiva, permitiéndoles acceder al contenido auditivo de manera fácil y entendible para ellos. El software contará con funcionalidades de reconocimiento de texto, traducción precisa a imágenes animadas y una interfaz de usuario amigable. Se prevé soporte de español a lenguaje de señas e integración con plataformas de comunicación, el proyecto requerirá un cronograma claro para llevarlo a cabo, así como un presupuesto definido para desarrollo, pruebas.

Funcionalidades

- **Traducir audio a lenguaje de señas:** Esta funcionalidad permite que el software traduzca en tiempo real audios hablados a lenguaje de señas. Utilizará algoritmos avanzados de reconocimiento de voz para convertir el discurso en texto y luego generará una representación visual animada del texto en lenguaje de señas.
- **Guardar audios en “Favoritos”:** Los usuarios podrán marcar ciertos audios traducidos como favoritos para acceder a ellos rápidamente en el futuro.
- **Retomar audio:** Esta funcionalidad permite a los usuarios continuar la reproducción de un audio desde donde lo dejaron anteriormente.
- **Estado del audio:** Muestra información sobre la duración del audio, el progreso de la traducción y otros detalles relevantes.
- **Guardar un audio con un nombre:** Permite a los usuarios asignar nombres descriptivos a los audios traducidos para una mejor identificación.
- **Etiquetar audios:** Los usuarios pueden agregar etiquetas o categorías personalizadas a los audios traducidos para facilitar su organización y búsqueda.
- **Exportar audios:** Permite a los usuarios asignar nombres descriptivos a los audios traducidos para una mejor identificación.
- **Importar audios:** Esta función permite a los usuarios cargar archivos de audio desde su dispositivo para su traducción.

Funcionalidades Futuras

- **Traducir diferentes idiomas a lenguaje de señas:** El software admitirá la traducción de una variedad de idiomas hablados al lenguaje de señas.
- **Traducción por medio de imágenes a lenguaje de señas:** Además de la traducción de texto a lenguaje de señas, el software generará imágenes animadas para representar conceptos y frases de manera más visual y comprensible.

Definiciones y Acrónimos

SL: Sign Language (Lenguaje de Señas)

LSE: Lengua de Signos Española

TTS: Text-to-Speech (Texto a Voz)

STT: Speech-to-Text (Voz a Texto)

UI: User Interface (Interfaz de Usuario)

UX: User Experience (Experiencia de Usuario)

Herramienta de software: Aplicación informática diseñada para facilitar una tarea específica.

Lenguaje de señas: Sistema de comunicación visual y gestual utilizado por personas sordas o con discapacidad auditiva.

Accesibilidad: Característica de un producto, servicio o entorno que permite su uso por parte de todas las personas, incluidas aquellas con discapacidad.

Reconocimiento de texto: Proceso de convertir texto impreso o manuscrito en formato digital.

Traducción precisa: Proceso de convertir un texto de un idioma a otro de manera exacta y fiel.

Interfaz de usuario: Conjunto de elementos que permiten la interacción entre un usuario y un sistema informático.

Cronograma: Planificación detallada de las actividades y plazos de un proyecto.

Desarrollo: Proceso de creación de un producto o servicio.

Pruebas: Evaluación del funcionamiento de un producto o servicio.

Descripción General

Objetivos del Sistema

El objetivo del sistema es proporcionar una herramienta de software que traduce audios en tiempo real a lenguaje de señas mediante imágenes animadas, facilitando el acceso al contenido auditivo para personas sordas o con discapacidad auditiva. Esta herramienta educativa y de accesibilidad se diseñará con el propósito de mejorar la comunicación y la inclusión, ofreciendo una plataforma intuitiva y versátil para la traducción precisa y visual de audios. El software contará con funcionalidades avanzadas de reconocimiento de voz, traducción a lenguaje de señas, y una interfaz de usuario amigable. Además, se prevé el soporte de español a lenguaje de señas y la integración con plataformas de comunicación.

Conceptos de las entidades

En el desarrollo de software educativo para la traducción de lenguaje de señas se refiere a la representación de objetos, personas o ideas con significado dentro del sistema. Las entidades organizan y estructuran la información, permitiendo traducir texto a imágenes animadas que representan signos específicos del lenguaje de señas. Estas entidades tienen atributos visuales y pueden relacionarse con otras partes del sistema para facilitar la traducción en tiempo real. El uso adecuado de entidades garantiza una herramienta educativa eficaz para personas sordas o con discapacidad auditiva, permitiéndoles acceder a contenido de manera adaptada a sus necesidades.

Funcionalidad General

Usuarios del Sistema

Los siguientes usuarios pueden interactuar con la pizarra dependiendo de las funcionalidades.

Funcionalidad	Administradores	Docente	Alumno	Invitado
Traducir audio a lenguaje de señas		✓	✓	✓
Guardar audios en "Favoritos"	✓	✓		
Retomar audio	✓	✓		
Estado del audio		✓		
Guardar un audio con un nombre	✓	✓		
Etiquetar audios	✓	✓	✓	
Exportar audios	✓	✓	✓	
Importar audios	✓	✓	✓	

Restricciones

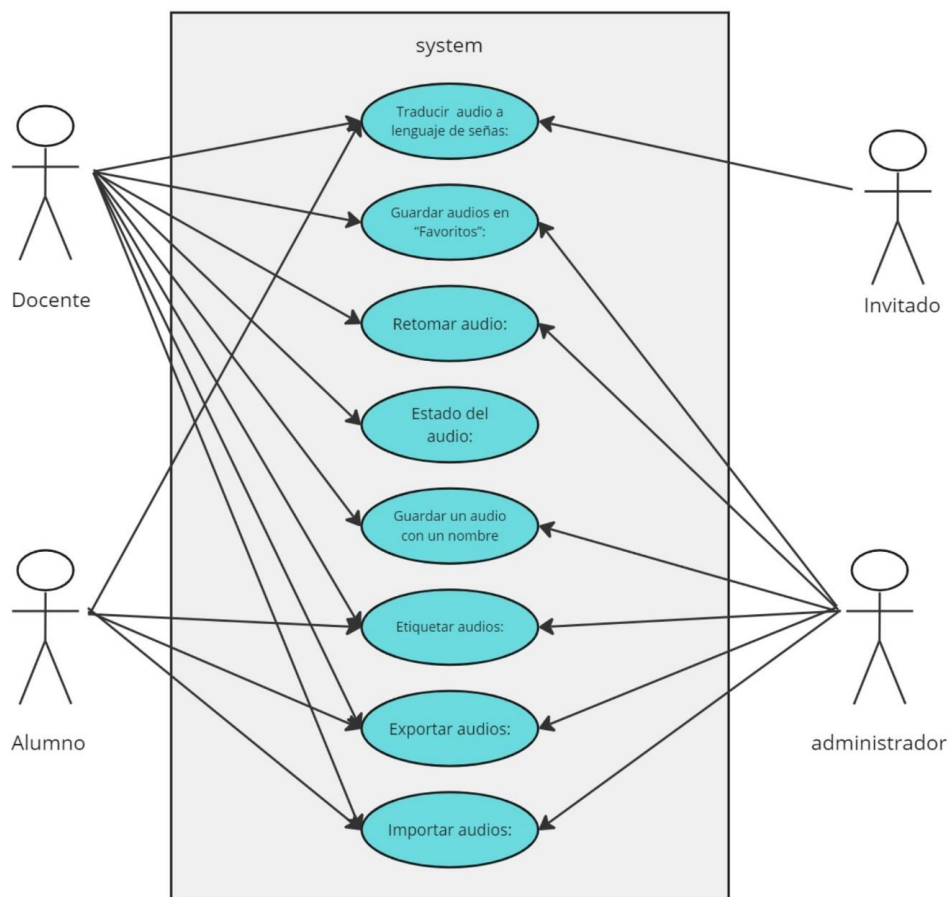
Invitados: No tienen acceso a la mayoría de las funcionalidades principales debido a las restricciones de usuario para asegurar que solo los usuarios registrados (administradores, docentes y alumnos) puedan utilizar las funciones avanzadas del software.

Docentes y Administradores: Tienen acceso a la mayoría de las funcionalidades, permitiéndoles administrar y utilizar el software de manera efectiva.

Alumnos: Pueden utilizar funcionalidades clave que les permiten interactuar con el contenido traducido, como traducir, etiquetar, exportar e importar audios.

Requisitos Funcionales

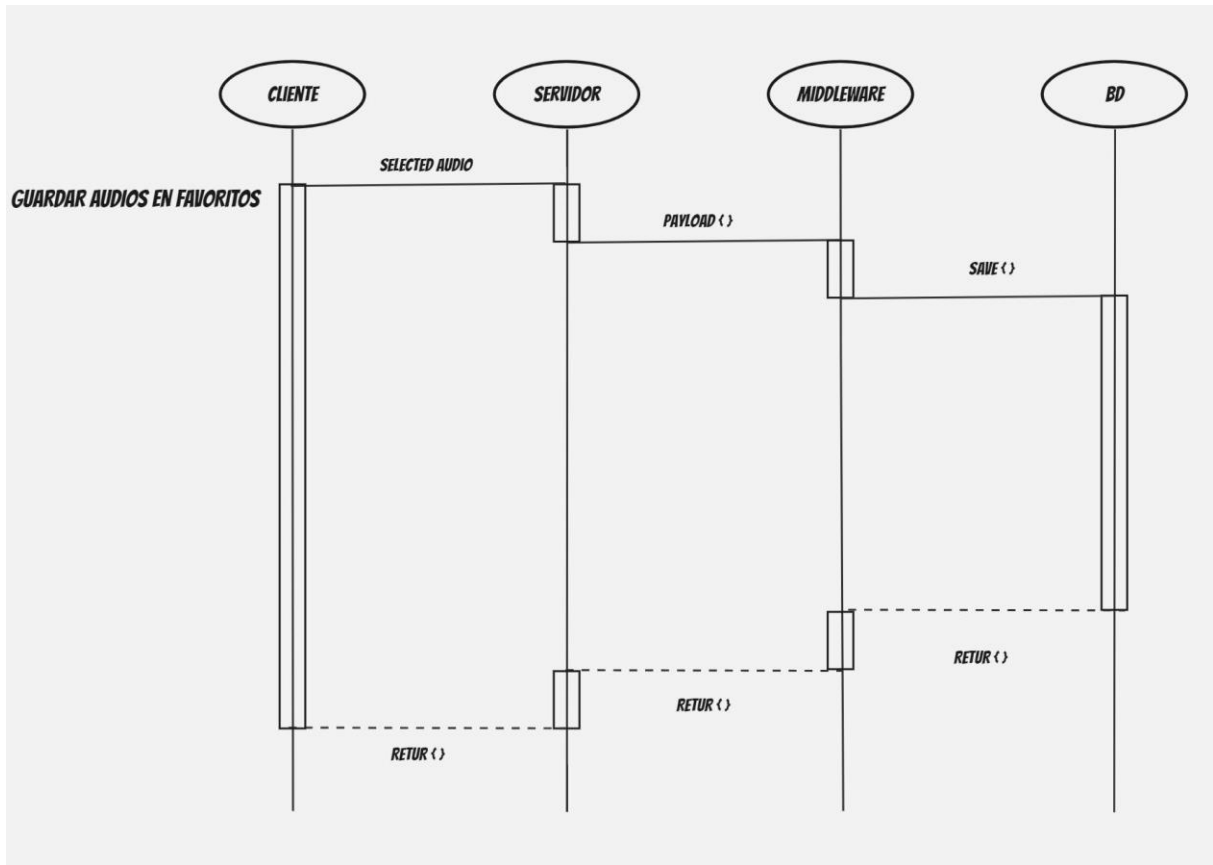
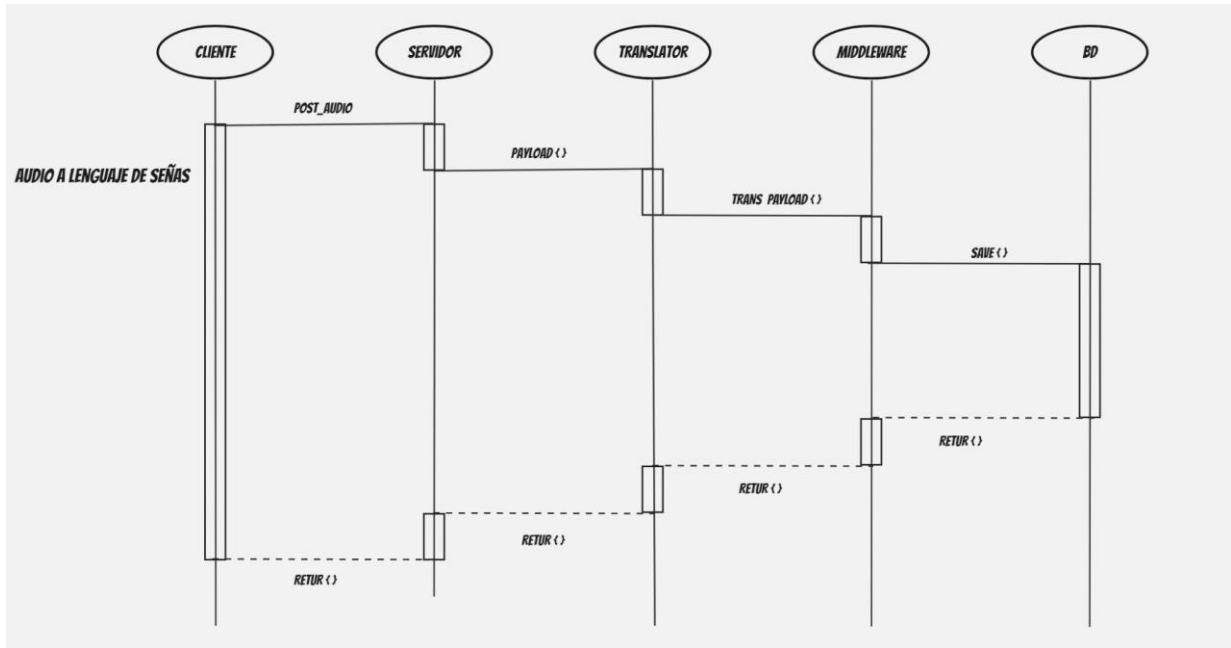
Casos de Uso

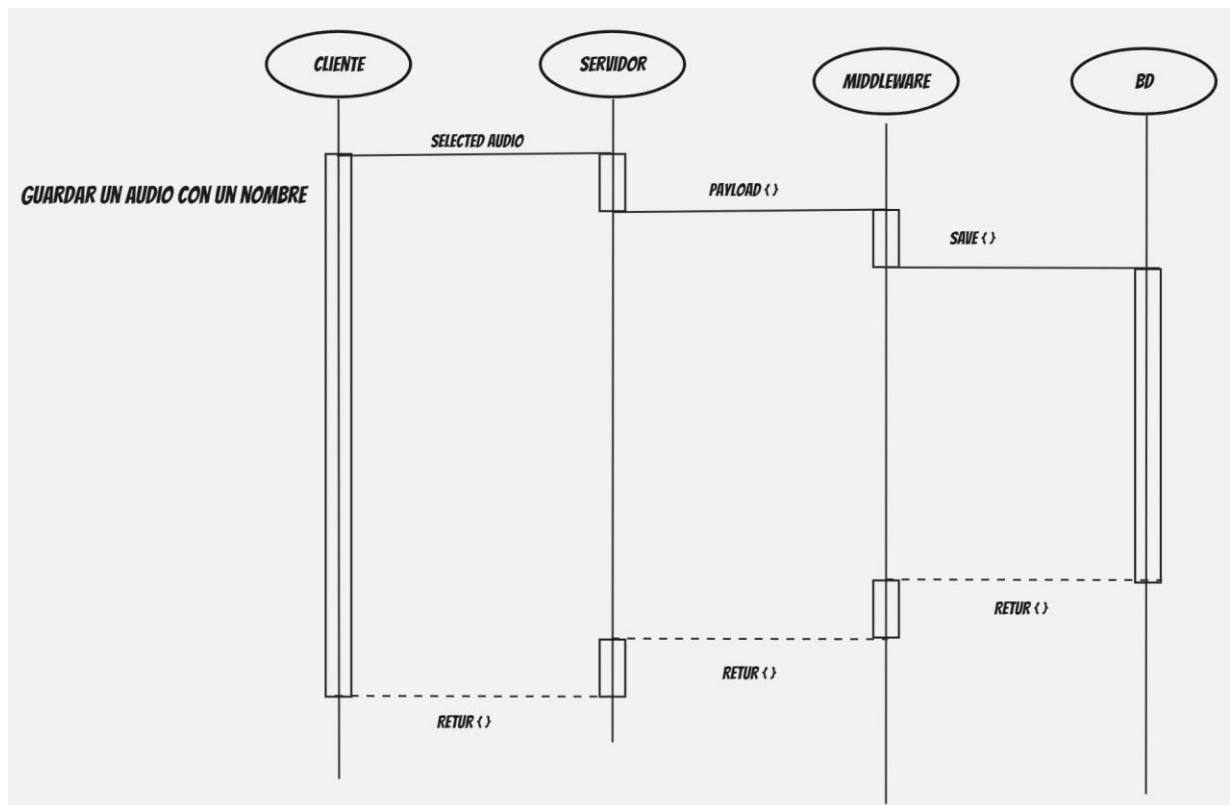
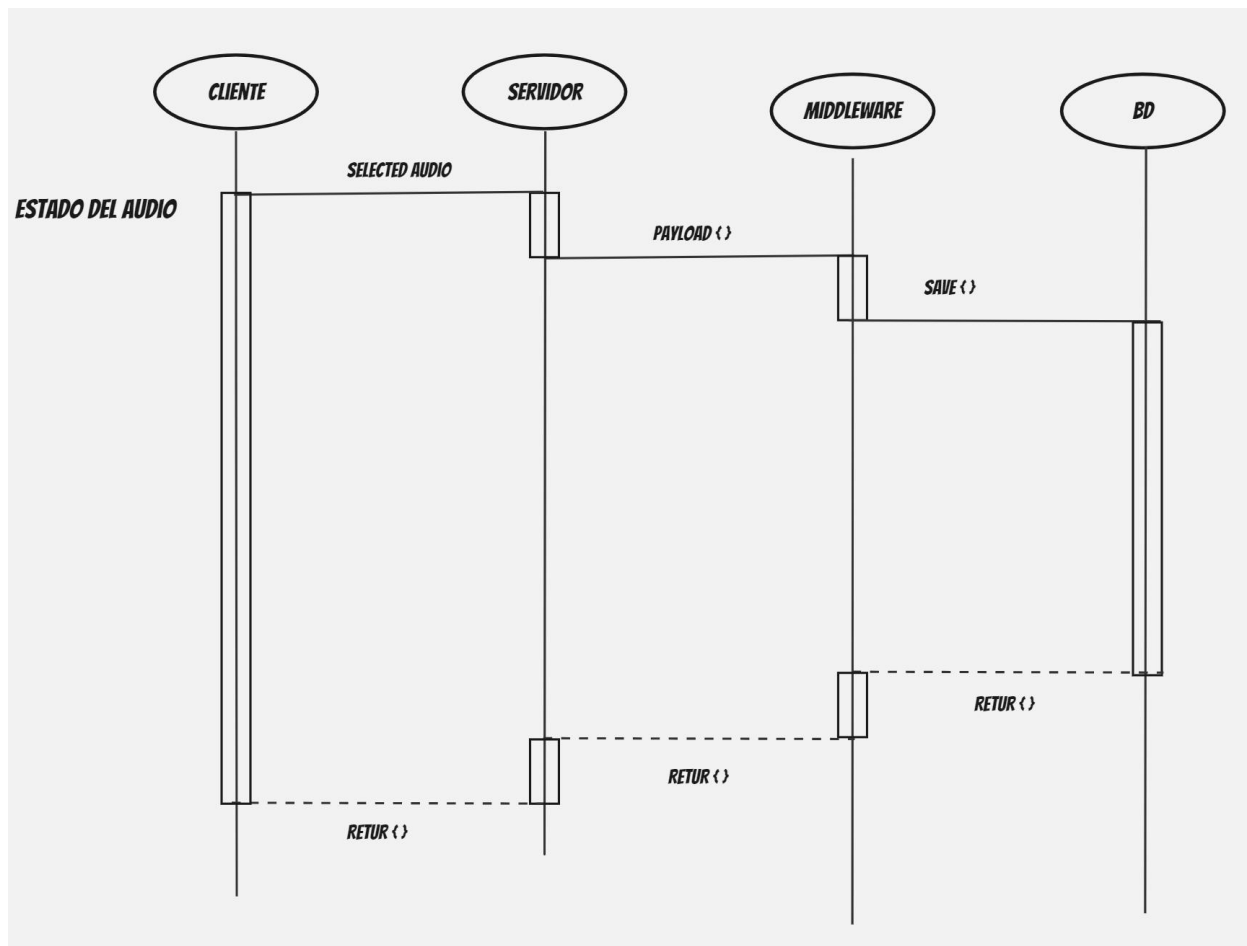


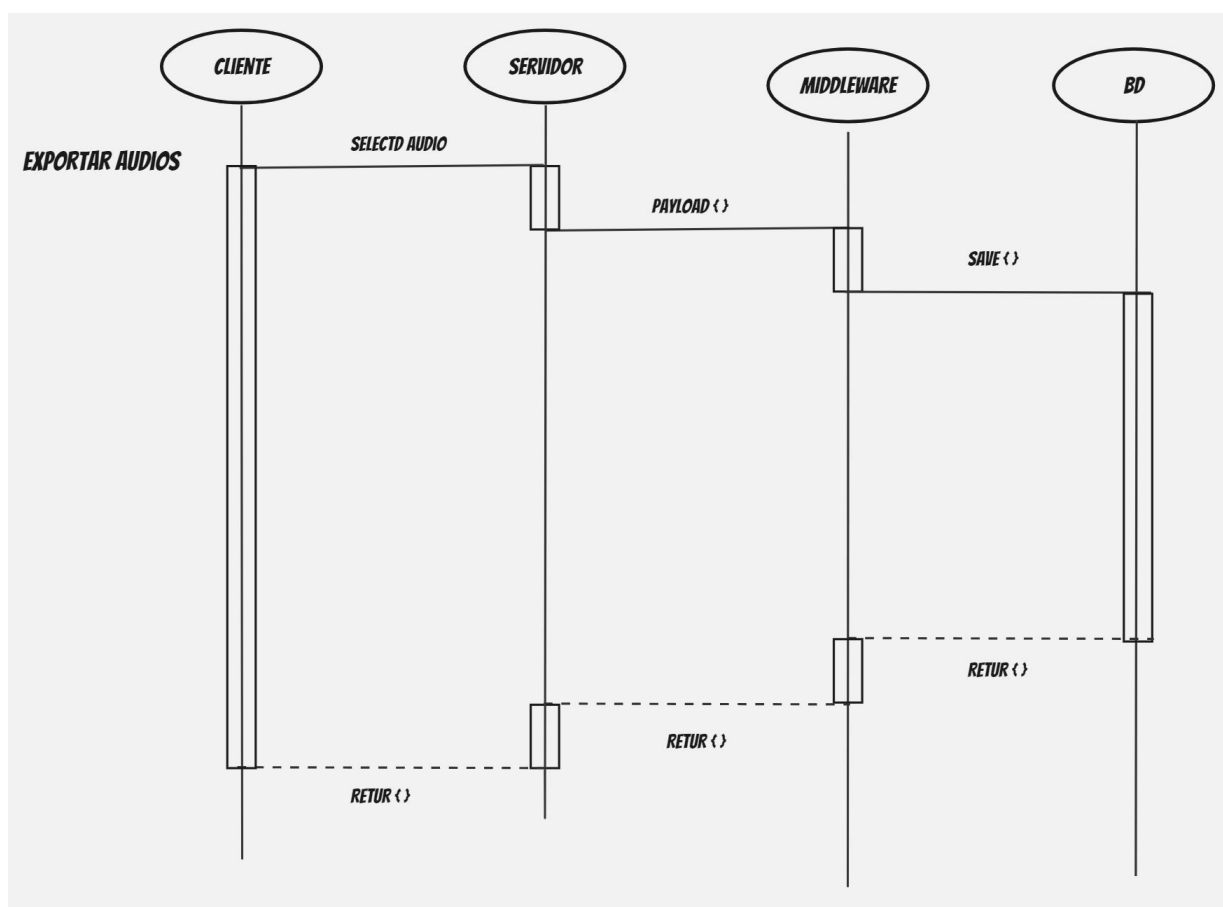
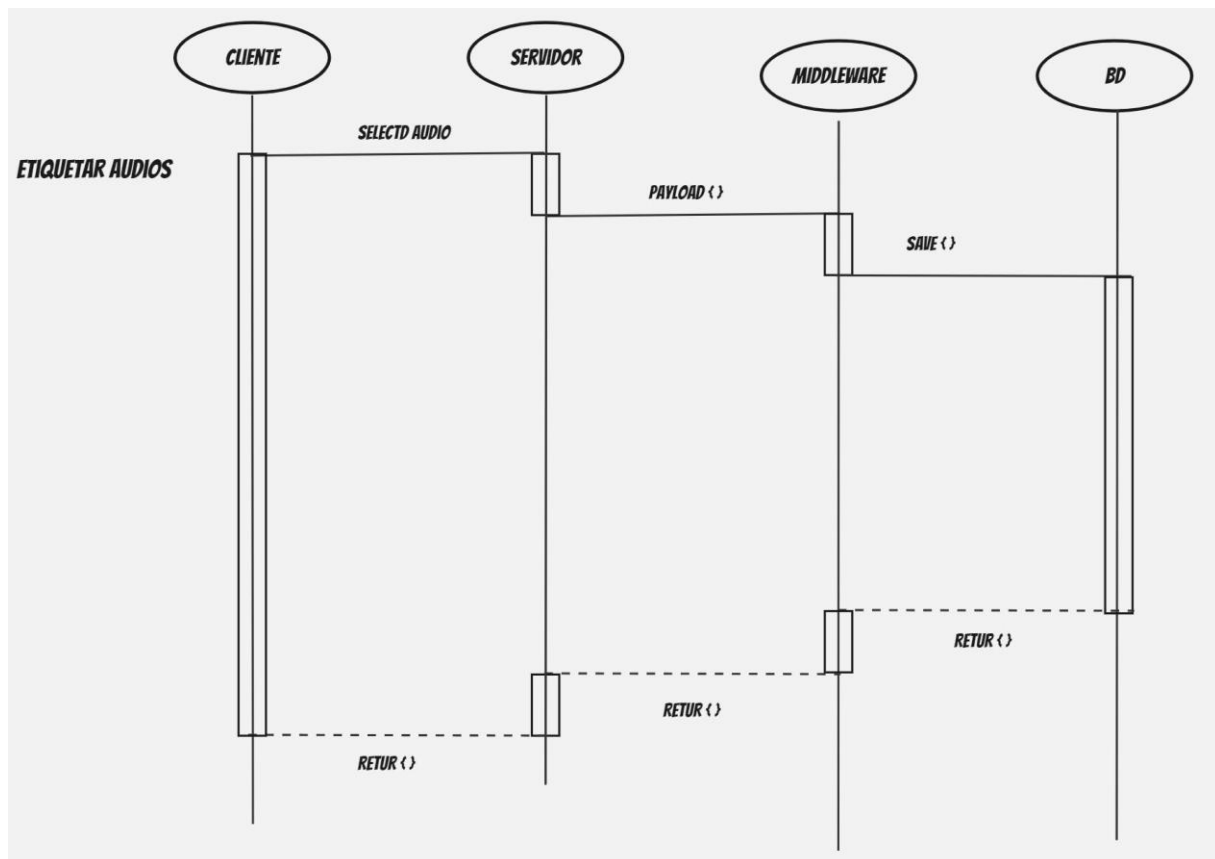
Descripción detallada de cada caso de uso

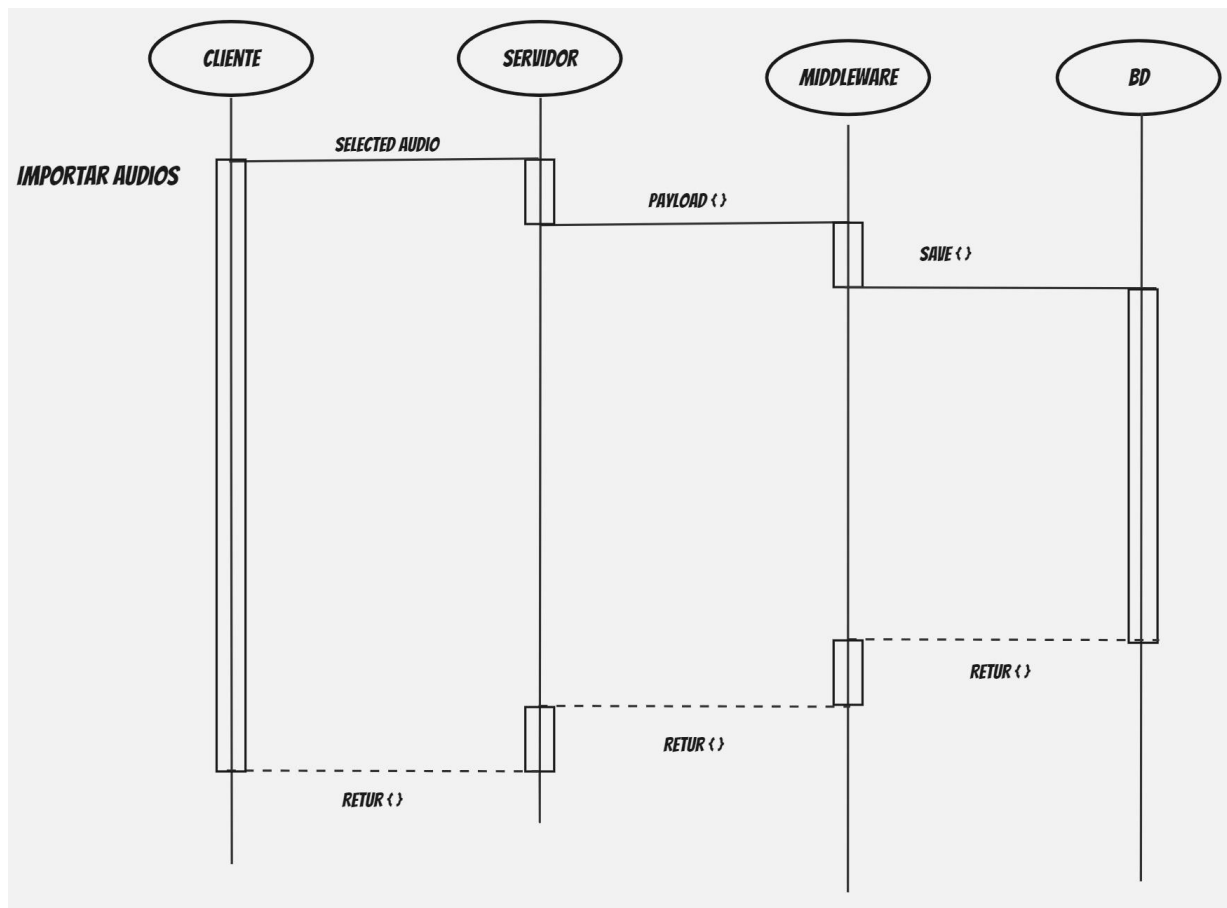
Diagramas de Flujo de Casos de Uso

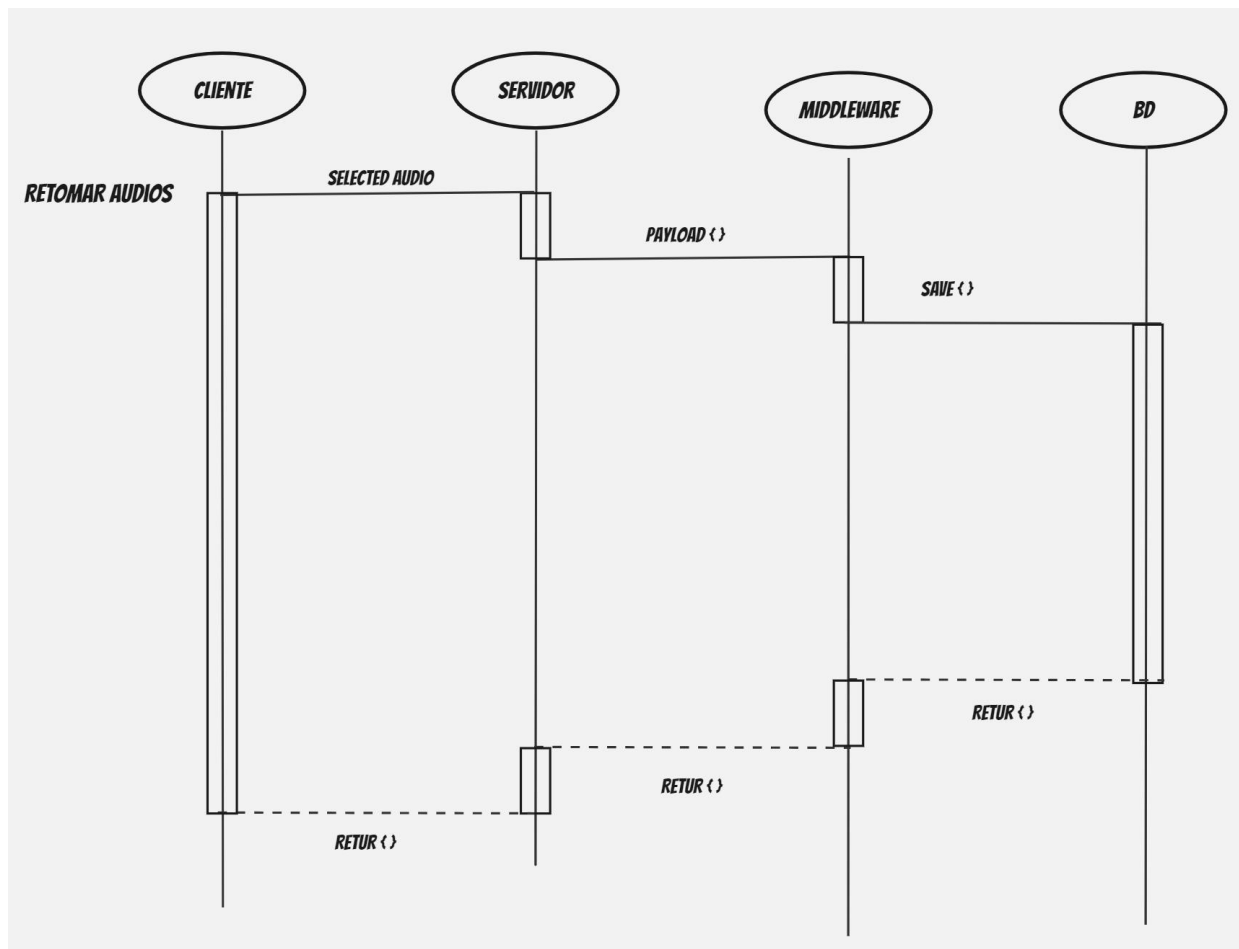
Diagramas de Secuencia











Prioridad de Requisitos

Requisitos No Funcionales

Seguridad:

- La pizarra debe garantizar la seguridad de los datos y la autenticación de usuarios. Debe utilizar cifrado para proteger la información.

Rendimiento:

- La aplicación debe ofrecer un rendimiento óptimo, permitiendo la colaboración en tiempo real incluso con un gran número de usuarios.

Escalabilidad:

- La pizarra debe ser escalable para manejar un aumento en el número de usuarios y la cantidad de contenido.

Disponibilidad:

- La aplicación debe estar disponible y funcionando de manera constante, minimizando el tiempo de inactividad.

Compatibilidad con Dispositivos:

- La pizarra debe ser compatible con una variedad de dispositivos, incluyendo computadoras de escritorio, tabletas y teléfonos móviles.

Usabilidad:

- La interfaz de usuario de la pizarra debe ser intuitiva y fácil de usar para usuarios de diferentes niveles de habilidad.

Accesibilidad:

- La aplicación debe ser accesible para personas con discapacidades, cumpliendo con estándares de accesibilidad web.

Cumplimiento Normativo:

- La pizarra debe cumplir con regulaciones y normativas de privacidad y seguridad de datos.

Tiempo de Respuesta:

- La aplicación debe tener tiempos de respuesta rápidos para mantener una experiencia de usuario fluida.

Requisitos de Desempeño

Velocidad de traducción:

El sistema debe ser capaz de procesar y traducir el audio en tiempo real o con un retraso mínimo. Se espera una velocidad de traducción cercana al tiempo real, con un máximo de 1 a 2 segundos de latencia entre el reconocimiento del audio y la presentación de la imagen correspondiente en lenguaje de señas.

Uso de memoria:

El software debe ser eficiente en el uso de memoria, evitando sobrecargas que afecten el rendimiento. El consumo de memoria debe mantenerse dentro de los límites aceptables para dispositivos de gama media, no superando los 500 MB en uso estándar, aunque puede variar dependiendo de la complejidad del procesamiento de las imágenes.

Almacenamiento:

El almacenamiento necesario debe ser escalable dependiendo del número de imágenes y modelos de aprendizaje utilizados para la traducción. Se recomienda que el sistema requiera un mínimo de 5 GB de almacenamiento para los recursos de imágenes y modelos. También debe tener la capacidad de actualizar y expandir la base de datos de imágenes sin afectar significativamente el almacenamiento.

Tiempo de respuesta:

El sistema debe responder de manera rápida a las solicitudes de traducción de los usuarios. Se espera un tiempo de respuesta de 1 a 3 segundos entre la entrada del audio y la visualización del signo correspondiente en la pantalla.

📌 **Tiempo de exportación e importación:**

Las funciones de exportación e importación de datos, como imágenes o grabaciones de audio para mejorar el sistema, deben ejecutarse en un tiempo razonable. Para archivos de audio e imágenes de tamaño medio (5-10 MB), se espera que las operaciones de exportación o importación no excedan los 5 segundos.

📌 **Ancho de banda:**

En el caso de que el sistema funcione en la nube o dependa de una conexión a internet, el ancho de banda requerido debe ser bajo para no afectar a usuarios con conexiones limitadas. Se estima un consumo de 1 Mbps para la transmisión de audio e imágenes, pero debe optimizarse para adaptarse a diferentes condiciones de red.

📌 **Capacidad de concurrencia de usuarios:**

El sistema debe poder manejar múltiples usuarios simultáneos sin que esto afecte el desempeño. Se recomienda que soporte al menos 100 usuarios concurrentes en un entorno de red estable, manteniendo la calidad y velocidad de la traducción.

Requisitos de Seguridad

📌 **Acceso seguro:**

Implementar autenticación segura (contraseñas fuertes y autenticación multifactor) y usar cifrado (HTTPS/TLS) para proteger las sesiones de usuario.

📌 **Protección de datos:**

Cumplir con las normativas de privacidad (como GDPR) y evitar la captura de datos personales sin consentimiento.

📌 **Auditoría y registro de traducciones:**

Registrar todas las traducciones y actividades del sistema, accesibles solo para administradores autorizados, y almacenarlas de forma segura por un tiempo determinado.

📌 **Control de versiones:**

Mantener un sistema de control de versiones que registre los cambios y garantice actualizaciones seguras sin afectar el rendimiento del sistema.

📌 **Variables y entorno:**

Cifrar las variables de configuración y aislar los entornos de desarrollo y producción para evitar filtraciones y errores.

📌 **Almacenamiento seguro:**

Proteger la información almacenada y las bases de datos con controles de acceso estrictos.

Requisitos de Usabilidad

📌 **Interfaz intuitiva:**

El diseño del software será limpio y sencillo para que cualquier usuario, independientemente de su nivel de experiencia, pueda navegar fácilmente por las opciones y herramientas disponibles.

📺 Minimización de pasos:

El flujo de trabajo será optimizado para que los usuarios puedan grabar y gestionar sus archivos con el menor número de pasos posibles, facilitando un uso eficiente y rápido.

📺 Accesibilidad:

Se incluirán características de accesibilidad, como atajos de teclado y opciones para usuarios con discapacidades, asegurando que el software sea accesible para todos los usuarios.

📺 Coherencia visual y funcional:

El diseño de la plataforma será consistente en todos los aspectos visuales y funcionales, permitiendo que los usuarios se familiaricen rápidamente con la estructura y el uso de las herramientas.

📺 Interacción fluida:

Se garantizará que la respuesta del sistema a las acciones del usuario sea rápida y coherente, evitando retrasos o comportamientos inesperados que puedan generar confusión o frustración.

Requisitos de Escalabilidad

📺 Adaptabilidad al crecimiento de usuarios:

La plataforma soportará un aumento en el número de usuarios simultáneos sin degradar el rendimiento, manteniendo la velocidad y calidad de las traducciones.

📺 Escalabilidad de funcionalidades:

La arquitectura será modular, permitiendo agregar nuevas características o mejoras sin afectar las funcionalidades existentes ni interrumpir el servicio.

📺 Gestión eficiente de recursos:

Se optimizará el uso de CPU, memoria y almacenamiento para manejar una mayor carga de trabajo, asegurando un desempeño constante conforme crezca la demanda.

📺 Soporte para diferentes resoluciones y calidades de imagen:

La plataforma gestionará eficientemente imágenes y videos en distintas resoluciones y calidades, garantizando que la experiencia del usuario no se vea comprometida por la variabilidad en los archivos procesados

Modelado E/R



ARCHIVOS:	AUDIOS:	PALABRAS:	SIGNIFICADO:
-ID: ARCHIVO 156	-NOMBRE: MATERIA 1	-TEXTO: HOLA	-TEXTO: HOLA
-URL: TAREA45	-URL: MATERIA1AUDIO2566	-REPRESENTACION GRAFICA:	- SEÑAL:
-TAMAÑO:20MG	-DURACION: 10MINUTO		
-FORMATO: MP3	-FECHA DE CREACION:21/09/2024		
	-ETIQUETAS: ESTUDIOS		
	-FAVORITOS: SI		
	-ESTADO: VALIDO		
	-ID AUDIO: AUDIO5461		
	-TIEMPO_PAUSA: 2MINUTOS		

Diagrama de Entidad-Relación

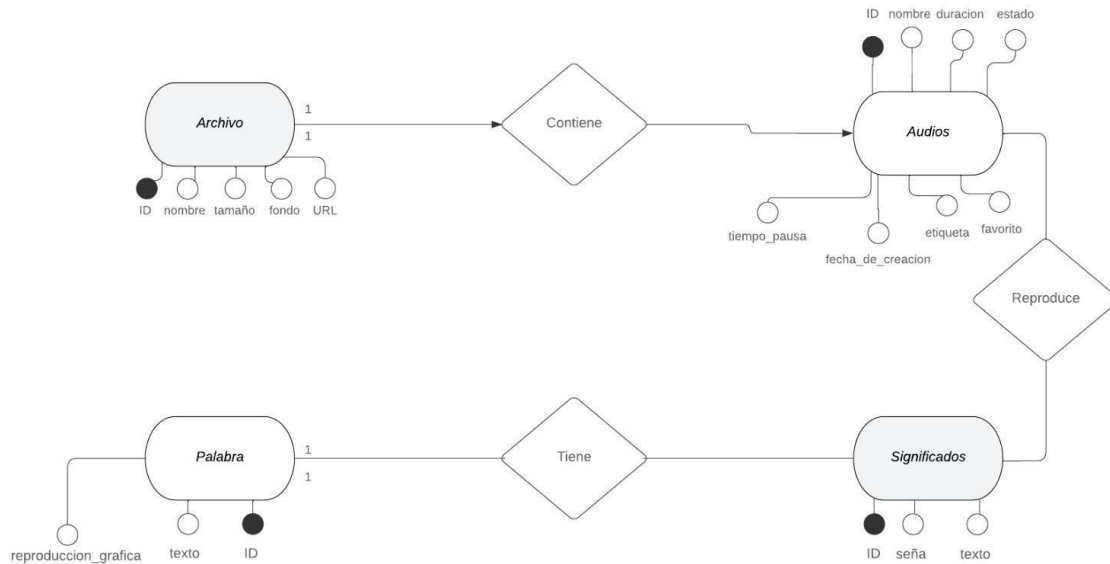


Diagrama relacional



Etapa 2: Persistencia de Datos con Backend

Introducción

Propósito de la Etapa

En esta etapa se lleva a cabo el proceso de persistencia de datos del proyecto sign_translator, donde se crea el diseño del backend de datos, conexión a bases de datos y las demás interacciones que permiten el funcionamiento correcto del software, para proyectarlo a la realización del frontend el semestre siguiente.

Alcance de la Etapa

El alcance de esta etapa de nuestro proyecto sign translator se centra en establecer una base sólida para el correcto manejo y almacenamiento de datos, esenciales para el correcto funcionamiento de software. lo que incluye el diseño y desarrollo del backend, la implementación de la arquitectura de la base de datos, y la creación de conexiones seguras y eficientes entre el backend y la base de datos. además, se contemplan todas las interacciones necesarias para garantizar la integridad y accesibilidad de los datos. este trabajo es fundamental para asegurar que, en el próximo semestre, el desarrollo del frontend pueda llevarse a cabo de manera fluida y efectiva, basándose en un sistema de backend robusto y bien estructurado.

Definiciones y Acrónimos

CRUD: Representa las operaciones básicas de interacción con una base de datos, esenciales para gestionar datos:

- **Create:** Crear un nuevo registro.
- **Read:** Leer o recuperar datos existentes.
- **Update:** Modificar registros existentes.
- **Delete:** Eliminar registros.

Node.js: Es un entorno de ejecución para JavaScript, diseñado para construir aplicaciones del lado del servidor. Permite a los desarrolladores usar JavaScript para programar tanto en el frontend como en el backend, y es conocido por su alto rendimiento en aplicaciones de red.

Mongo Atlas: Es una plataforma de base de datos en la nube para MongoDB. Facilita la gestión de bases de datos al ofrecer almacenamiento, escalabilidad y seguridad automatizadas, permitiendo a los desarrolladores centrarse en sus aplicaciones sin preocuparse por la administración de la base de datos.

Middleware: En el contexto de aplicaciones web y APIs, el middleware es una función o conjunto de funciones que se ejecutan entre la solicitud del cliente y la respuesta del servidor.

Se utiliza para tareas como la autenticación, el manejo de errores, el procesamiento de datos, o el registro de solicitudes.

Controller: componentes que procesan solicitudes ODBS y devuelven datos a la aplicación. Si es necesario, los controladores modifican la solicitud de una aplicación en un formulario comprendido por el origen de datos

Service: sistema software diseñado para soportar la interacción máquina-a-máquina, a través de una red, de forma interoperable.

Repository: lugar de almacenamiento del cual pueden ser recuperados e instalados los paquetes de software en un ordenador.

Interceptor: objeto que permite instrumentar la aplicación para que capture datos de interés.

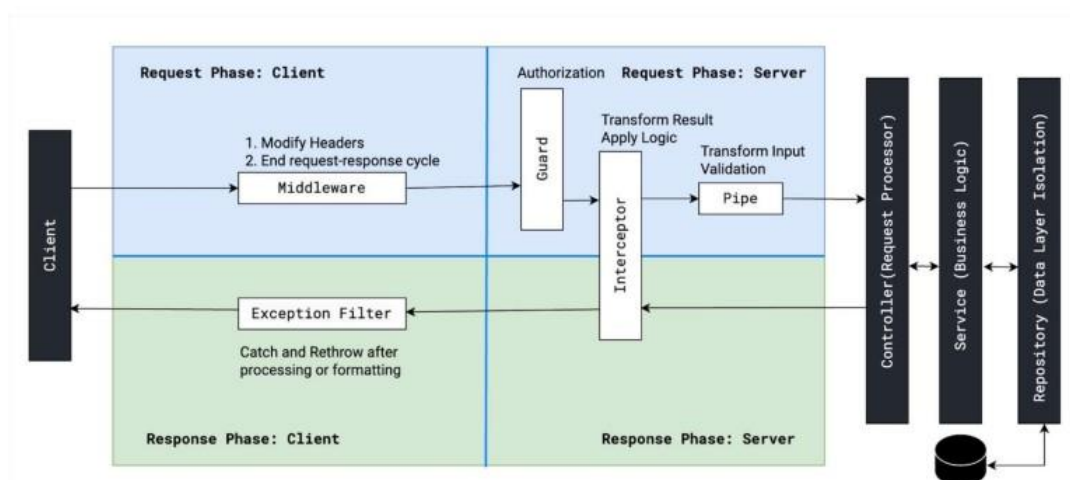
Server: sistema que proporciona recursos, datos, servicios o programas a otros ordenadores, conocido como clientes, a través de una red.

Diseño de la Arquitectura de Backend

Descripción de la Arquitectura Propuesta

Este proyecto está basado en un sistema de Nest.js que cuenta con múltiples elementos que incluyen cliente, el interceptor que proporciona una validación al controlador y existe una interacción entre el controlador, servidor y el repositorio, al igual que con la base de datos.

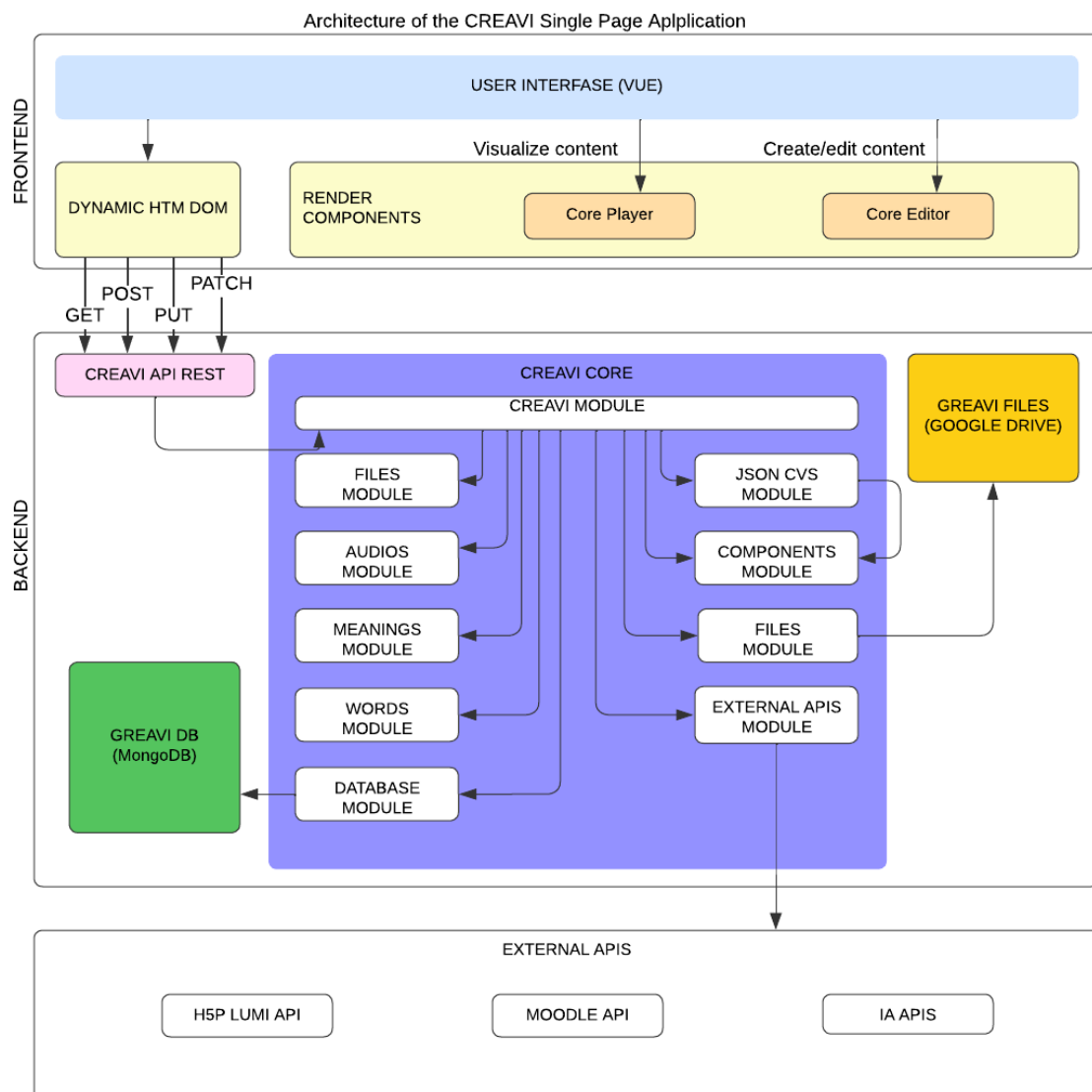
Componentes del backend



En esta etapa del desarrollo backend, avanzamos desde el *Service (Business Logic)* capa donde se maneja la lógica de negocio, donde se procesan y transforman los datos que se reciben del

repositorio antes de enviarlos al controlador. También se pueden gestionar DTOS en esta capa para asegurar que los datos sean transferidos de manera correcta entre el cliente y el servidor. Hasta la *Repository (Data layer Isolation)* donde se maneja la conexión a la base de datos, las consultas, las operaciones CRUD (Create, Read, Update, Delete) y el uso de esquemas para estructuras de datos. En esta capa se encapsula toda la lógica relacionada con el acceso y la persistencia de datos.

Diagramas de Arquitectura

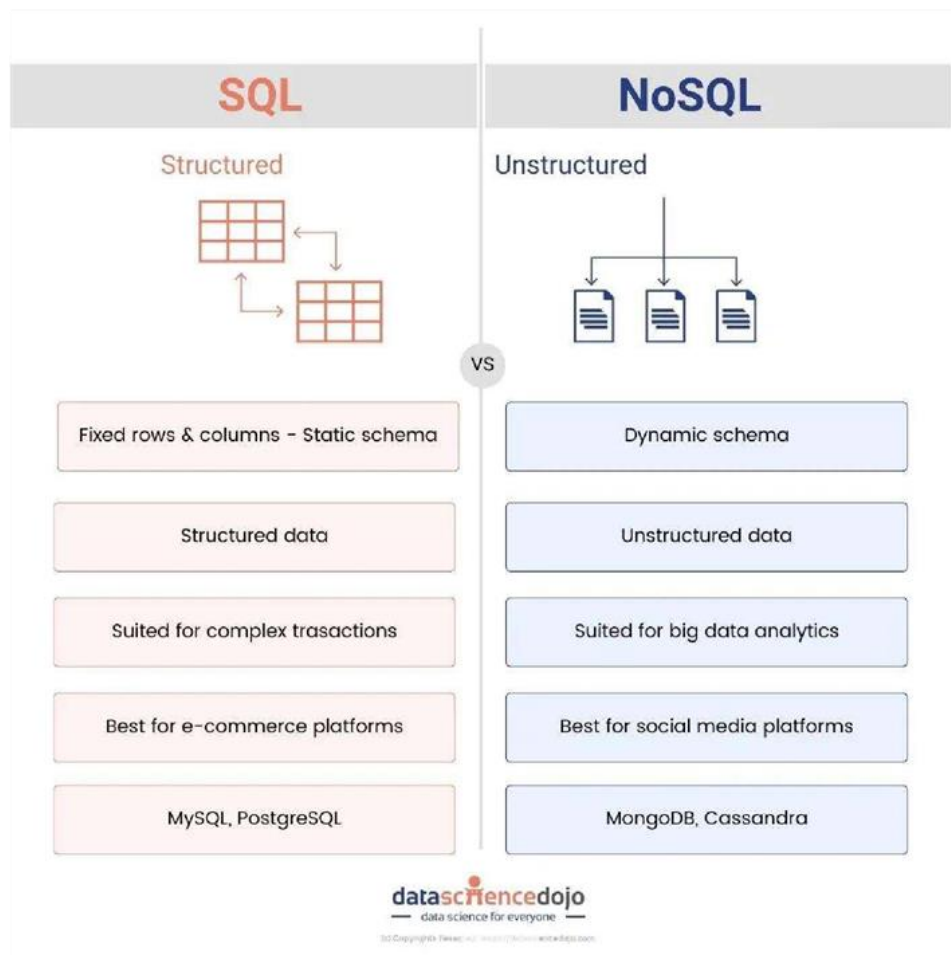


El diagrama anterior muestra la arquitectura de la aplicación de una pagina (SPA) de CREAVI, compuesta por una interfaz de usuario que incluye componentes para la visualización y edición de contenido, un backend que maneja la lógica de la aplicación mediante módulos, y una base de datos MongoDB. En esta arquitectura, se integrará el módulo sign translator, un sistema virtual que permitirá la traducción de audios a lenguaje de señas por medio de imágenes, sign translator Utiliza APIS de reconocimiento de voz para captar comandos, procesarlos y proporcionar las imágenes que se encargan de mostrar la traducción, En el backend, sign translator se añadirá como un nuevo módulo dentro de CREAVI.

Elección de la Base de Datos

Evaluación de Opciones (SQL o NoSQL)

Una de las decisiones más importantes para el desarrollo del backend del proyecto, es la base de datos que se usarán para el software. Entre los tipos de bases de datos más populares están las bases de datos SQL y NoSQL, la elección de una u otra, se basa en las necesidades que presente el software. En la siguiente tabla se muestran las principales diferencias entre SQL y NoSQL.



Justificación de la Elección

La elección de una base de datos NoSQL, como MongoDB, para este proyecto se justifica principalmente por su capacidad para manejar datos no estructurados y semiestructurados, lo cual es esencial en un sistema de traducción de audio a lenguaje de señas que debe almacenar audios, traducciones, imágenes animadas y otros metadatos asociados. A diferencia de una base de datos relacional, NoSQL ofrece flexibilidad al permitir almacenar diferentes tipos de datos

sin la necesidad de un esquema rígido, adaptándose así a la naturaleza diversa de los datos en el proyecto. Además, las bases de datos NoSQL como MongoDB están diseñadas para escalar horizontalmente, lo cual permite distribuir los datos en varios servidores, asegurando que el sistema pueda crecer en usuarios y volumen de información sin comprometer su rendimiento. Esto es particularmente importante en el contexto de un proyecto de accesibilidad donde el tiempo de respuesta es crucial para mantener una experiencia de usuario fluida y eficiente. Otro aspecto clave es que MongoDB utiliza un modelo de documentos basado en JSON o BSON, lo cual facilita el almacenamiento de datos jerárquicos y relacionales en un solo lugar, permitiendo organizar y acceder a todos los detalles de un audio traducido de forma sencilla y directa. Esta estructura de almacenamiento es mucho más eficiente para los requisitos de un sistema como el presente, en el cual es necesario manejar datos complejos y facilitar su recuperación en tiempo real. Asimismo, MongoDB y otras bases de datos NoSQL se integran fácilmente con tecnologías modernas de backend como Node.js, Express y plataformas en la nube, lo cual simplifica el desarrollo e implementación de una arquitectura moderna y escalable.

Diseño de Esquema de Base de Datos



Implementación del Backend

Elección del Lenguaje de Programación

Utilizar Node.js, Nest.js y Express.js en este software tiene varias ventajas. Con Node.js ofrece eficiencia y manejo asíncrono crucial para aplicaciones en tiempo real. Express.js facilita la creación de APIs robustas y seguras, ideal para gestionar solicitudes y respuestas del sistema y Nest.js proporciona una arquitectura modular y escalable integrando TypeScript para mejorar la mantenibilidad del código. Por lo tanto combinación con una base de datos SQL y MongoDB permiten flexibilidad en el almacenamiento y optimización del rendimiento. Esta tecnología conjunta asegura un desarrollo eficiente y accesible para los usuarios con necesidades especiales.

Conexión a la Base de Datos

Configuración de la Conexión

La conexión a la base de datos de MongoDB Atlas desde el componente Sing_translator se da mediante las credenciales de acceso, la URL de la conexión y una configuración específica relacionada con la seguridad del sistema. Al trabajar con mongo se deben instalar las dependencias con el siguiente comandando npm i @nestjs/mongoose mongoose

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { MongooseModule } from '@nestjs/mongoose';
import { ArchivosModule } from './archivos/archivos.module';
import { AudiosModule } from './audios/audios.module';
import { SignificadosModule } from './significados/significados.module';
import { PalabrasModule } from './palabras/palabras.module';
```

Una vez completado el proceso de instalación, podemos importar el Mongoose Module en la raíz AppModule. Se configuró el import que estaba en @module para obtener el enlace y mediante este se conectara a la base de datos en MongoDB Atlas en el Clusters

```
@Module({
  imports: [
    MongooseModule.forRoot(
      'mongodb+srv://jvegalomineh10[REDACTED].1vujg.mongodb.net/translator_db?retryWrites=true&w=majority&appName=Cluster0',
    ),
  ],
})
```

Desarrollo de Operaciones CRUD

Las operaciones CRUD (crear, leer, actualizar y editar) son fundamentales para el funcionamiento óptimo del componente y para interactuar con las colecciones en MongoDB, aquí se detalla cómo se implementan estas operaciones para las colecciones

Controller

```
import { Controller, Get, Post, Body, Patch, Param, Delete } from '@nestjs/common';
import { ArchivosService } from '../archivos.service';
import { CreateArchivoDto } from '../dto/create-archivo.dto';
import { UpdateArchivoDto } from '../dto/update-archivo.dto';

@Controller('archivos')
export class ArchivosController {
  constructor(private readonly archivosService: ArchivosService) {}

  @Post()
  create(@Body() createArchivoDto: CreateArchivoDto) {
    return this.archivosService.create(createArchivoDto);
  }

  @Get()
  findAll() {
    return this.archivosService.findAll();
  }

  @Get(':id')
  findOne(@Param('id') id: string) {
    return this.archivosService.findOne(id);
  }

  @Patch(':id')
  update(@Param('id') id: string, @Body() updateArchivoDto: UpdateArchivoDto) {
    return this.archivosService.update(id, updateArchivoDto);
  }

  @Delete(':id')
  remove(@Param('id') id: string) {
    return this.archivosService.remove(id);
  }
}
```

Service

```
import { Injectable } from '@nestjs/common';
import { CreateArchivoDto } from '../dto/create-archivo.dto';
import { UpdateArchivoDto } from '../dto/update-archivo.dto';
import { InjectModel } from '@nestjs/mongoose';
import { Model } from 'mongoose';
import { archivos } from '../schema/archivos.schema';

@Injectable()
export class ArchivosService {
  constructor(@InjectModel(archivos.name) private archivosModel: Model<archivos>) {}

  async create(createArchivoDto: CreateArchivoDto) {
    const createdarchivos = new this.archivosModel(createArchivoDto);
    const result = await createdarchivos.save();
    return result;
  }

  async findAll() {
    return this.archivosModel.find();
  }

  async findOne(id: string) {
    return this.archivosModel.findById(id);
  }

  async update(id: string, updateArchivoDto: UpdateArchivoDto) {
    try{
      const updatearchivo = await this.archivosModel.findByIdAndUpdate(
        id, updateArchivoDto, {new: true}
      );
      return updatearchivo
    }
  }
}
```

```
export class ArchivosService {
  async update(id: string, updateArchivoDto: UpdateArchivoDto) {
    try{
      const updatearchivo = await this.archivosModel.findByIdAndUpdate(
        id, updateArchivoDto, {new: true}
      );
    }
    finally{
      console.log("actualizado");
    }
  }

  async remove(id: string) {
    try{
      const deletearchivo = await this.archivosModel.findByIdAndDelete(id);
      return deletearchivo;
    }
    finally{
      //
    }
  }
}
```

Controller

```
import { Controller, Get, Post, Body, Patch, Param, Delete } from '@nestjs/common';
import { AudiosService } from '../audios.service';
import { CreateAudioDto } from '../dto/create-audio.dto';
import { UpdateAudioDto } from '../dto/update-audio.dto';

@Controller('audios')
export class AudiosController {
  constructor(private readonly audiosService: AudiosService) {}

  @Post()
  create(@Body() createAudioDto: CreateAudioDto) {
    return this.audiosService.create(createAudioDto);
  }

  @Get()
  findAll() {
    return this.audiosService.findAll();
  }

  @Get(':id')
  findOne(@Param('id') id: string) {
    return this.audiosService.findOne(id);
  }

  @Patch(':id')
  update(@Param('id') id: string, @Body() updateAudioDto: UpdateAudioDto) {
    return this.audiosService.update(id, updateAudioDto);
  }

  @Delete(':id')
  remove(@Param('id') id: string) {
    return this.audiosService.remove(id);
  }
}
```

Service

```
import { Injectable } from '@nestjs/common';
import { CreateAudioDto } from '../dto/create-audio.dto';
import { UpdateAudioDto } from '../dto/update-audio.dto';
import { InjectModel } from '@nestjs/mongoose';
import { Model } from 'mongoose';
import { audios } from '../schema/audios.schema';

@Injectable()
export class AudiosService {
  constructor(@InjectModel(audios.name) private audiosModel: Model<audios>) {}

  async create(createAudioDto: CreateAudioDto) {
    const createdaudios = new this.audiosModel(createAudioDto);
    const result = await createdaudios.save();
    return result;
  }

  findAll() {
    return this.audiosModel.find();
  }

  findOne(id: string) {
    return this.audiosModel.findById(id);
  }

  async update(id: string, updateAudioDto: UpdateAudioDto) {
    try{
      const updateaudio = await this.audiosModel.findByIdAndUpdate(
        id, updateAudioDto, {new: true}
      );
      return updateaudio
    }
  }
}
```

```
    finally{
      console.log("actualizado");
    }
  }

  async remove(id: string) {
    try{
      const deleteaudio = await this.audiosModel.findByIdAndDelete(id)
      return deleteaudio;
    }
    finally{
    }
  }
}
```

Main:

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';
import { SwaggerTheme, SwaggerThemeNameEnum } from 'swagger-themes'

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

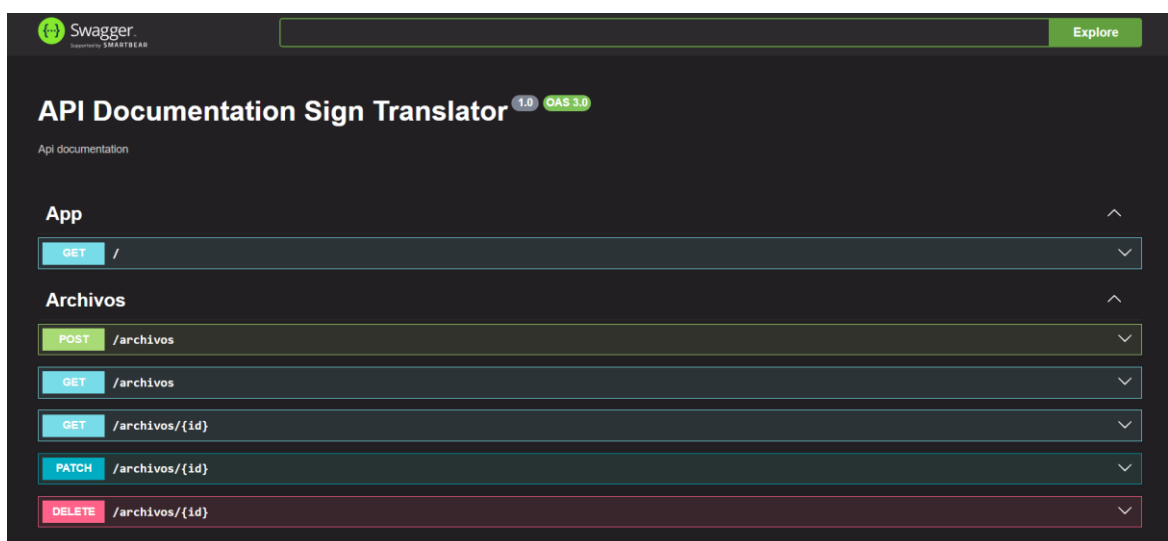
  const config = new DocumentBuilder()
    .setTitle("API Documentation Sign Translator")
    .setDescription("Api documentation")
    .setVersion("1.0")
    .build()


  const documentFactory = () => SwaggerModule.createDocument(app, config);

  const theme = new SwaggerTheme();

  const options = {
    explorer:true,
    customCss: theme.getBuffer(SwaggerThemeNameEnum.DARK_MONOKAI)
  }

  SwaggerModule.setup('api', app, documentFactory, options)
  await app.listen(process.env.PORT ?? 3000);
}
bootstrap();
```



Audios		^
POST	/audios	▼
GET	/audios	▼
GET	/audios/{id}	▼
PATCH	/audios/{id}	▼
DELETE	/audios/{id}	▼
Significados		^
POST	/significados	▼
GET	/significados	▼ 
GET	/significados/{id}	▼
PATCH	/significados/{id}	▼
DELETE	/significados/{id}	▼

Palabras

POST

/palabras

GET

/palabras

GET

/palabras/{id}

PATCH

/palabras/{id}

DELETE

/palabras/{id}

Schemas

CreateArchivoDto

UpdateArchivoDto

CreateAudioDto

UpdateAudioDto

Etapas 3: Consumo de Datos y Desarrollo Frontend

Introducción

Propósito de la Etapa

Alcance de la Etapa

Definiciones y Acrónimos

Creación de la Interfaz de Usuario (UI)

Diseño de la Interfaz de Usuario (UI) con HTML y CSS

Consideraciones de Usabilidad

Maquetación Responsiva

Programación Frontend con JavaScript (JS)

Desarrollo de la Lógica del Frontend

Manejo de Eventos y Comportamientos Dinámicos

Uso de Bibliotecas y Frameworks (si aplicable)

Consumo de Datos desde el Backend

Configuración de Conexiones al Backend

Obtención y Presentación de Datos

Actualización en Tiempo Real (si aplicable)

Interacción Usuario-Interfaz

Manejo de Formularios y Validación de Datos

Implementación de Funcionalidades Interactivas

Mejoras en la Experiencia del Usuario

Pruebas y Depuración del Frontend

Diseño de Casos de Prueba de Frontend

Pruebas de Usabilidad

Depuración de Errores y Optimización del Código

Implementación de la Lógica de Negocio en el Frontend

Migración de la Lógica de Negocio desde el Backend (si necesario)

Validación de Datos y Reglas de Negocio en el Frontend

Integración con el Backend

Verificación de la Comunicación Efectiva con el Backend

Pruebas de Integración Frontend-Backend