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

Componente Gestor de Tareas

Ever Luis Guerrero Bustillo: everluisgb123@gmail.com
Jose Trugoth Guerrero:

Breve reseña

El sistema propuesto tiene como objetivo registrar de manera eficiente las prácticas pedagógicas de los docentes en formación. Este registro no solo permitirá almacenar los datos relacionados con las actividades pedagógicas realizadas, sino también gestionar la documentación y los archivos requeridos o utilizados durante las prácticas. La plataforma buscará centralizar toda la información relevante, facilitando su acceso y seguimiento. Además, se garantizará que los docentes puedan almacenar y organizar los recursos educativos, así como generar reportes que favorezcan la evaluación y mejora continua del proceso formativo. Este enfoque contribuirá a una gestión más organizada y eficiente, alineándose con las necesidades de la formación académica moderna.

Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

Introducción

En el contexto actual de la educación superior, la sistematización de los procesos de la práctica docente se presenta como un desafío clave para mejorar la formación de futuros profesionales, especialmente en campos tan dinámicos como la informática y los medios audiovisuales. Este trabajo responde a la necesidad de digitalizar y sistematizar las experiencias y actividades realizadas en las prácticas docentes de la Licenciatura en Informática y Medios Audiovisuales de la Universidad de Córdoba. La digitalización de estos procesos busca agilizar las tareas administrativas y alinearse con la política institucional de cero papeles. Para fundamentar esta iniciativa, se realizaron entrevistas a docentes y estudiantes para conocer sus necesidades y preferencias respecto a la gestión de la información. De estas entrevistas surgió la necesidad de mantener registros precisos y accesibles, que faciliten la revisión y el seguimiento del progreso académico, evitando la pérdida de datos y reduciendo esfuerzos duplicados. En este marco, se propone el desarrollo de un software que centralice y organice los registros de las prácticas docentes en una base de datos, integrando funcionalidades clave basadas en los requerimientos identificados. Este proyecto busca ser el punto de partida para crear una herramienta digital que optimice los procesos de formación y contribuya a la mejora continua de la enseñanza en la Universidad de Córdoba**.**

Propósito del Documento

El propósito de este proyecto es mejorar la eficiencia y efectividad del proceso educativo mediante una mejor organización, acceso y seguimiento de la información educativa. Este sistema permitirá facilitar la evaluación continua del progreso del alumnado y la adaptación de estrategias pedagógicas en tiempo real, promoviendo una educación más personalizada que responda a las necesidades individuales de cada estudiante.

El proyecto busca optimizar la organización, el seguimiento y la personalización del proceso educativo, estructurando de manera eficiente todas las actividades y recursos relacionados con la enseñanza. Esto permitirá un monitoreo constante del progreso estudiantil, mejorando la comunicación dentro de la comunidad educativa y adaptando las experiencias de aprendizaje a las necesidades específicas de estudiantes y docentes.

- 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, algebra 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

1. Este proyecto tiene como objetivo principal desarrollar un software que permita la digitalización y sistematización de las actividades y experiencias de las prácticas docentes en la Licenciatura en Informática y Medios Audiovisuales de la Universidad de Córdoba. El sistema está orientado a facilitar la gestión de la información, el seguimiento del progreso académico y la mejora de los procesos pedagógicos.

2. Límites del Proyecto

- Usuarios Involucrados:
    - Docentes tutores: Podrán asignar, supervisar y evaluar tareas de los docentes en formación.
    - Docentes en formación: Registrarán actividades, cargarán recursos educativos y recibirán retroalimentación.
    - Administradores: Configurarán roles, usuarios y parámetros del sistema.
- Áreas de Aplicación:
    - Gestión de planes de clase.
    - Seguimiento de evaluaciones, observaciones y estrategias pedagógicas.
    - Almacenamiento seguro de recursos educativos y registros de actividades.
- Exclusiones:
    - El proyecto no contempla el diseño de contenido académico, únicamente su gestión.
    - No incluirá módulos específicos para otros programas académicos fuera de la Licenciatura en Informática y Medios Audiovisuales.

3. Funcionalidades Clave

1. Creación y Asignación de Tareas:
    - Registro de actividades como planes de clase, observaciones y evaluaciones.
    - Asignación de tareas específicas a docentes en formación por parte de los tutores.
2. Gestión y Consulta de Información:
    - Búsqueda avanzada de tareas por palabras clave, categorías o estados.

- Visualización de tareas organizadas en listas y calendarios.
- Priorización y actualización del estado de las tareas.

3. Análisis y Evaluación:
   - Generación de informes sobre el rendimiento académico.
   - Retroalimentación directa entre tutores y docentes en formación.
4. Colaboración y Comunicación:
   - Espacios para comentarios y discusiones sobre tareas específicas.
   - Integración con plataformas externas como Google Drive y Google Calendar.
5. Seguridad y Privacidad:
   - Implementación de roles y permisos.
   - Cumplimiento con normativas de protección de datos institucionales.
6. Manejo de Archivos Adjuntos:
   - Carga y consulta de documentos, como formatos de práctica y recursos didácticos.

4. Resultados Esperados

- Mayor eficiencia en la gestión de las prácticas docentes.
- Reducción del uso de papel mediante la digitalización de los procesos.
- Mejor comunicación entre docentes tutores y docentes en formación.
- Acceso más ágil a los registros académicos en tiempo real.

5. Restricciones Técnicas

- El sistema debe ser accesible desde dispositivos móviles y computadoras.
- La interfaz debe ser intuitiva y usable para personas con diferentes niveles de habilidades tecnológicas.
- La base de datos debe permitir el almacenamiento seguro de grandes volúmenes de información.

6. Plazo y Entregables

- Duración Estimada: 6 meses para la implementación completa.
- Entregables:
  - Prototipo funcional del sistema.
  - Documentación técnica y manuales de usuario.
  - Capacitación para los usuarios finales.

Definiciones y Acrónimos

API: Interfaz de Programación de Aplicaciones (Application Programming Interface).

DBMS: Sistema de Gestión de Bases de Datos (Database Management System).

SQL: Lenguaje de Consulta Estructurada (Structured Query Language).

HTTP: Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol).

REST: Transferencia de Estado Representacional (Representational State Transfer).

JSON: Notación de Objetos de JavaScript (JavaScript Object Notation).

JWT: Token de Web JSON (JSON Web Token).

CRUD: Crear, Leer, Actualizar y Borrar (Create, Read, Update, Delete).

ORM: Mapeo Objeto-Relacional (Object-Relational Mapping).

MVC: Modelo-Vista-Controlador (Model-View-Controller).

API RESTful: API que sigue los principios de REST.

CI/CD: Integración Continua / Entrega Continua (Continuous Integration / Continuous Delivery).

SaaS: Software como Servicio (Software as a Service).

SSL/TLS: Capa de sockets seguros/Seguridad de la Capa de Transporte (Secure Sockets Layer/Transport Layer Security).

HTML: Lenguaje de Marcado de Hipertexto (Hypertext Markup Language).

CSS: Hojas de Estilo en Cascada (Cascading Style Sheets).

JS: JavaScript. DOM: Modelo de Objeto del Documento (Document Object Model).

UI: Interfaz de Usuario (User Interface). UX: Experiencia del Usuario (User Experience).

SPA: Aplicación de Página Única (Single Page Application).

AJAX: Asincrónico JavaScript y XML (Asynchronous JavaScript and XML).

CMS: Sistema de Gestión de Contenido (Content Management System).

CDN: Red de Distribución de Contenido (Content Delivery Network).

SEO: Optimización de Motores de Búsqueda (Search Engine Optimization).

IDE: Entorno de Desarrollo Integrado (Integrated Development Environment).

CLI: Interfaz de Línea de Comandos (Command Line Interface).

PWA: Aplicación Web Progresiva (Progressive Web App).

CRUD: Operaciones básicas de creación (Create), lectura (Read), actualización (Update) y eliminación (Delete) de datos en una base de datos.

AJAX: Asincrónico JavaScript y XML (Asynchronous JavaScript and XML). Permite actualizar partes de una página web sin necesidad de recargarla completamente, lo que mejora la experiencia del usuario.

OAuth: Protocolo de autorización que permite a una aplicación obtener acceso limitado a recursos en un servicio en nombre del propietario de los recursos, sin necesidad de compartir sus credenciales.

GUI: Interfaz Gráfica de Usuario (Graphical User Interface). Permite a los usuarios interactuar con el software a través de elementos visuales como ventanas, botones y menús.

GESTOR DE TAREAS: es una herramienta que te ayuda a organizar y administrar tus actividades diarias, proyectos y responsabilidades. Puede ser tanto una aplicación digital como una técnica de gestión personal.

INTERFAZ INTUITIVA: Una interfaz de usuario diseñada de manera que sea fácil de entender y utilizar, sin requerir una explicación extensa o capacitación previa por parte del usuario.

PERSONALIZABLE: La capacidad de adaptar o modificar una herramienta o aplicación de acuerdo con las preferencias y necesidades individuales de cada usuario.

DISEÑO CENTRADO EN EL USUARIO: Un enfoque de diseño que prioriza las necesidades, deseos y habilidades del usuario final durante el proceso de desarrollo de productos o servicios.

EFICIENCIA: La capacidad de realizar una tarea o alcanzar un objetivo utilizando la menor cantidad de recursos posibles, como tiempo, esfuerzo o dinero.

PRODUCTIVIDAD: La medida en que se logran los resultados deseados de manera efectiva y eficiente, maximizando el rendimiento y minimizando el desperdicio.

PLANIFICACIÓN: El proceso de establecer objetivos, identificar acciones y recursos necesarios, y establecer un cronograma para alcanzar esos objetivos de manera sistemática.

SEGUIMIENTO: La acción de monitorear y controlar el progreso de las tareas o actividades para asegurarse de que se estén llevando a cabo de acuerdo con el plan establecido.

INNOVACIÓN: La introducción de nuevas ideas, métodos o productos que generan cambios positivos y mejoras significativas en la forma en que se realizan las actividades o se satisfacen las necesidades.

EXPERIENCIA DEL USUARIO: La percepción general y la satisfacción del usuario al interactuar con un producto o servicio, que incluye aspectos como la facilidad de uso, la eficiencia y la utilidad percibida.

BACKEND: La parte de un sistema informático que se encarga del procesamiento y almacenamiento de datos, así como de la lógica de negocio, generalmente no visible para el usuario final.

PERSISTENCIA DE DATOS: La capacidad de almacenar y recuperar datos de manera permanente en un sistema informático, incluso después de que se haya cerrado la aplicación o se haya apagado el dispositivo.

Descripción General

Esta herramienta digital se concibe como un medio para optimizar la gestión de la información, permitiendo su consulta y actualización en tiempo real, tanto para docentes como para estudiantes. En este sentido, se explorará el uso de diversas herramientas de programación y bases de datos, con el objetivo de diseñar un software que se adapte de manera óptima a las necesidades específicas de la Licenciatura en Informática

Objetivos del Sistema

Crear y poner en marcha un sistema educativo digital completo que funcione como una innovadora herramienta para mejorar las prácticas pedagógicas fortaleciendo así las habilidades de los docentes y optimizar los procesos de enseñanza y aprendizaje fomentando el desarrollo de habilidades en los futuros profesionales de la educación

.

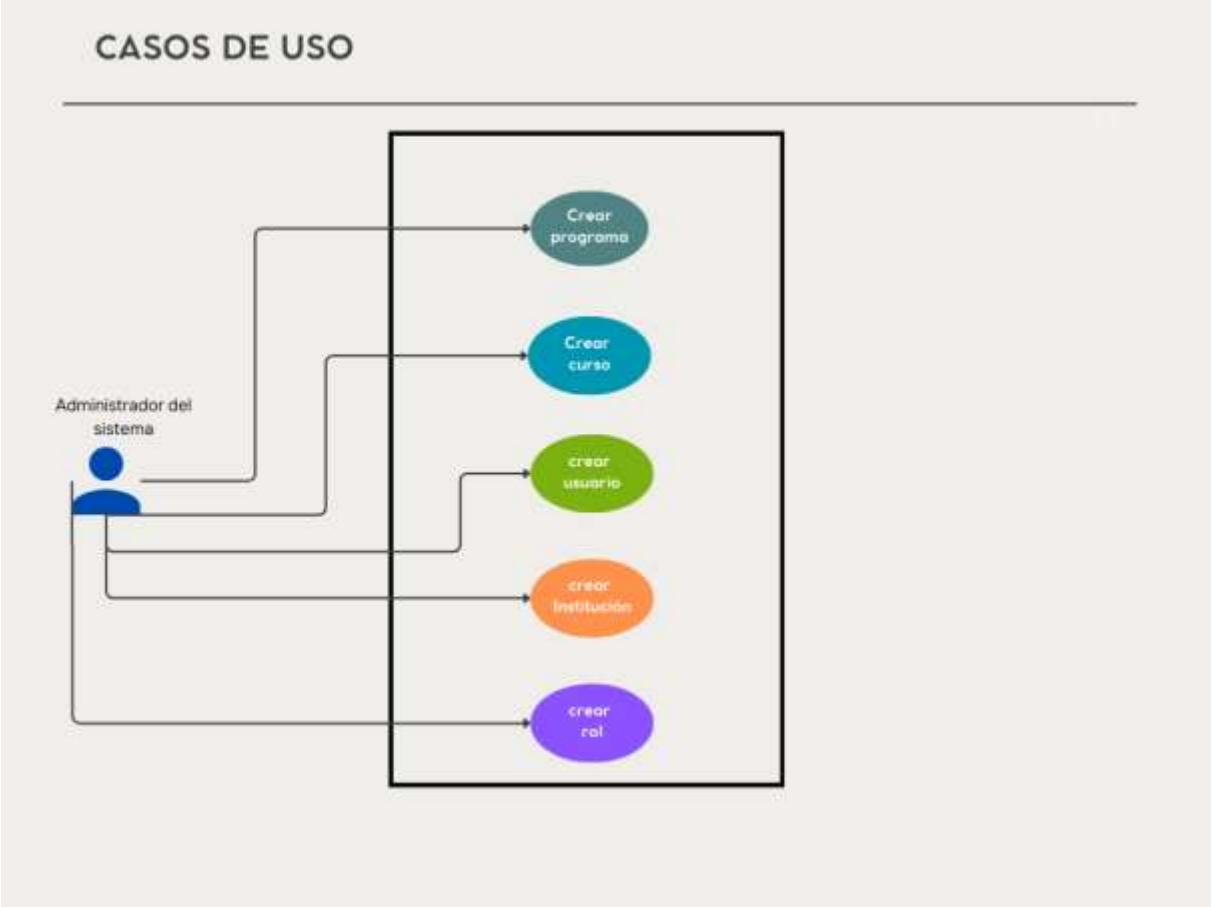
Funcionalidad General

El sistema permite la organización, registro, seguimiento y evaluación de todas las actividades relacionadas con las prácticas docentes, proporcionando herramientas para optimizar el trabajo tanto de los docentes tutores como de los docentes en formación.

Características Principales:

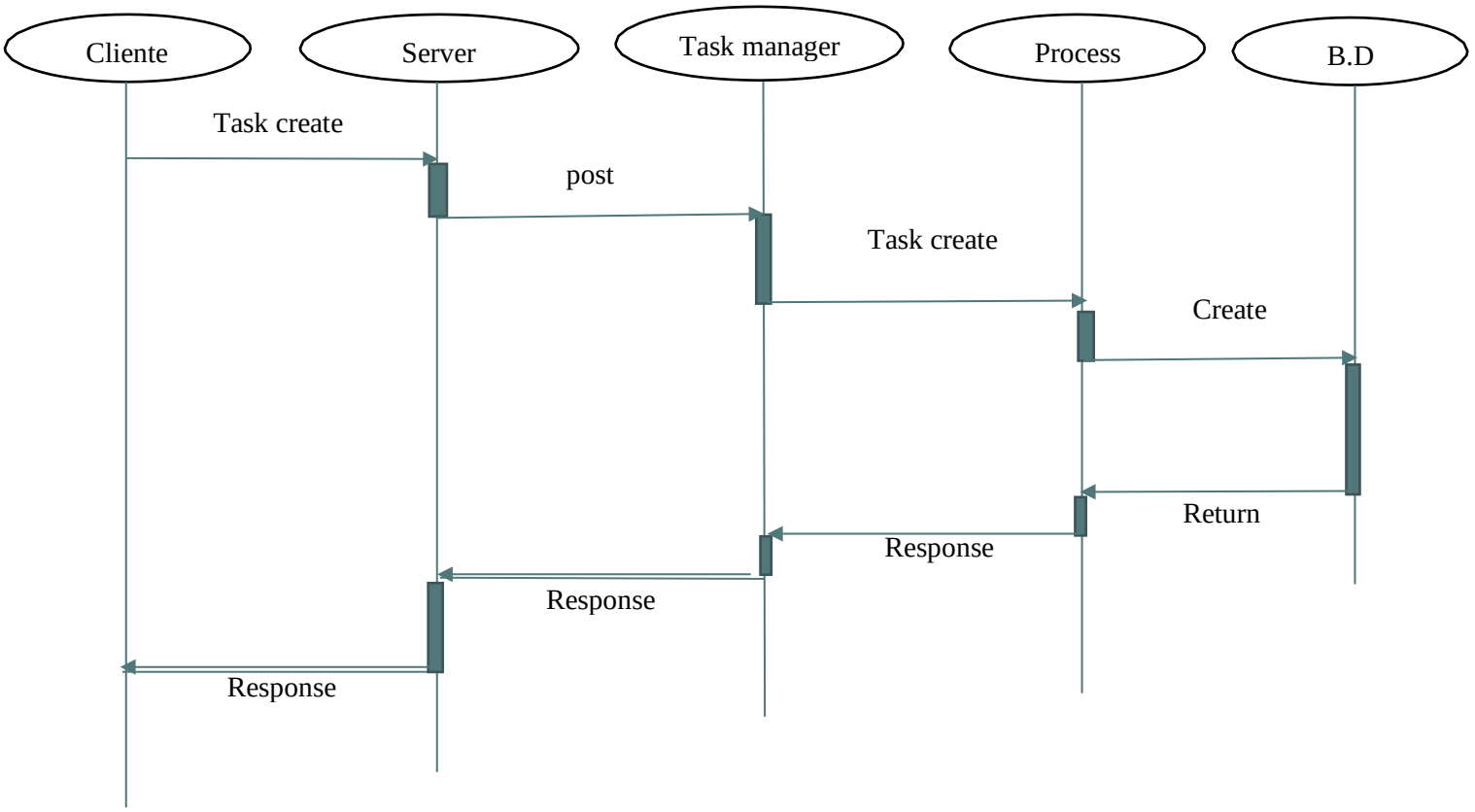
1. Registro y Gestión de Tareas:
    - Los docentes tutores pueden asignar tareas específicas (planes de clase, observaciones, evaluaciones) a los docentes en formación.
    - Los docentes en formación pueden registrar actividades realizadas y adjuntar archivos relacionados.
2. Seguimiento del Progreso Académico:
    - Permite registrar estados de las actividades (iniciado, en proceso, en revisión, terminado).
    - Visualización de las tareas pendientes, próximas a vencer o completadas.
3. Colaboración y Comunicación:
    - Espacios para comentarios y retroalimentación entre tutores y docentes en formación.
    - Notificaciones automáticas para alertar sobre fechas límite y cambios en las tareas.
4. Análisis y Evaluación:
    - Generación de informes sobre el desempeño de los docentes en formación.
    - Identificación de patrones y áreas de mejora en la práctica docente.
5. Almacenamiento y Seguridad:
    - Base de datos centralizada para el almacenamiento de documentos relevantes, como formatos de práctica, observaciones y evaluaciones.
    - Implementación de roles y permisos para proteger la privacidad y seguridad de los datos.
6. Integración y Accesibilidad:
    - Sincronización con herramientas externas como Google Drive y Google Calendar para facilitar el acceso y la planificación.
    - Acceso desde múltiples dispositivos (computadoras y móviles) con una interfaz intuitiva y fácil de usar.

Casos de Uso

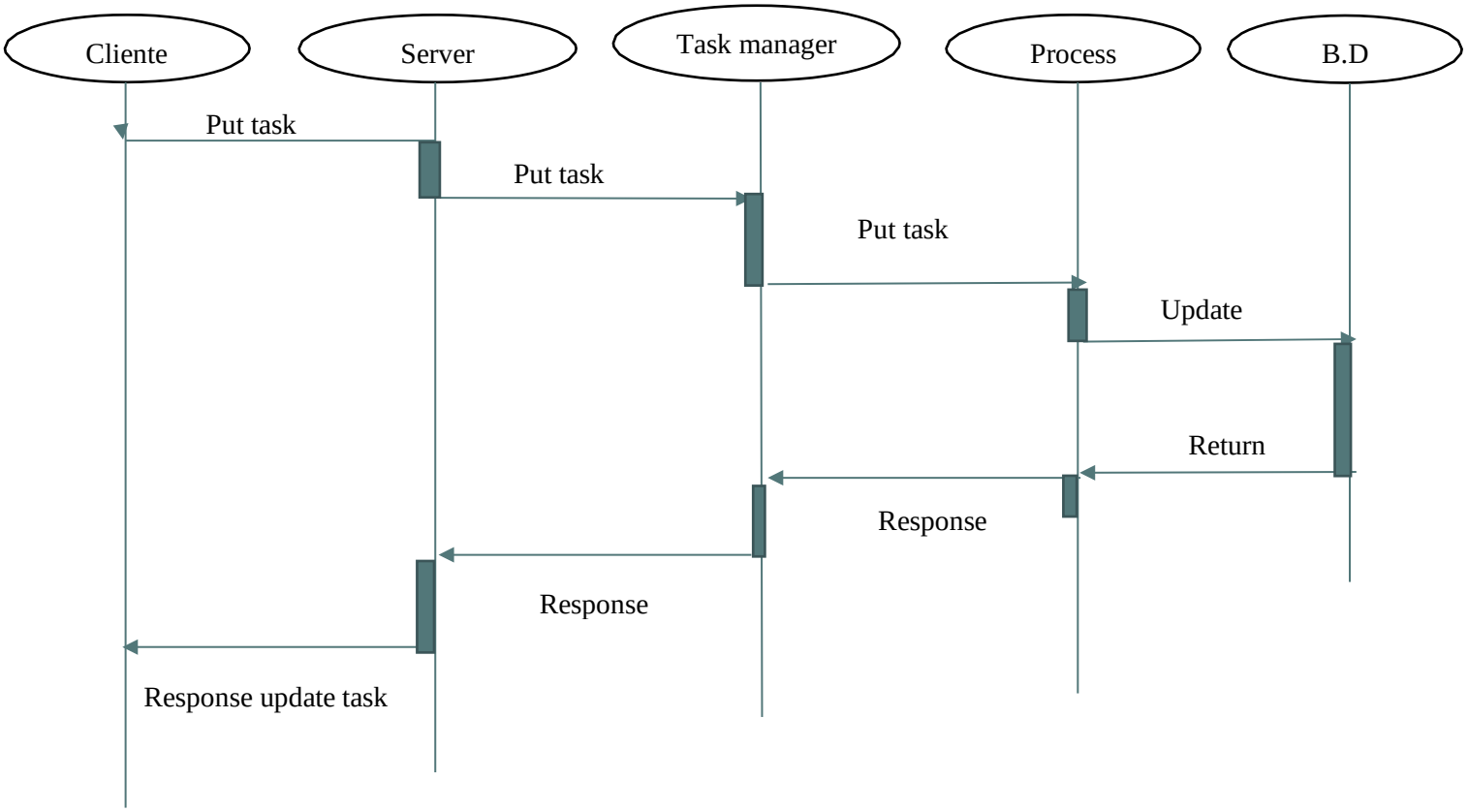

CASOS DE USO
Administrador del sistema
Crear programa
Crear curso
crear usuario
crear institución
crear rol

Diagramas de Secuencia

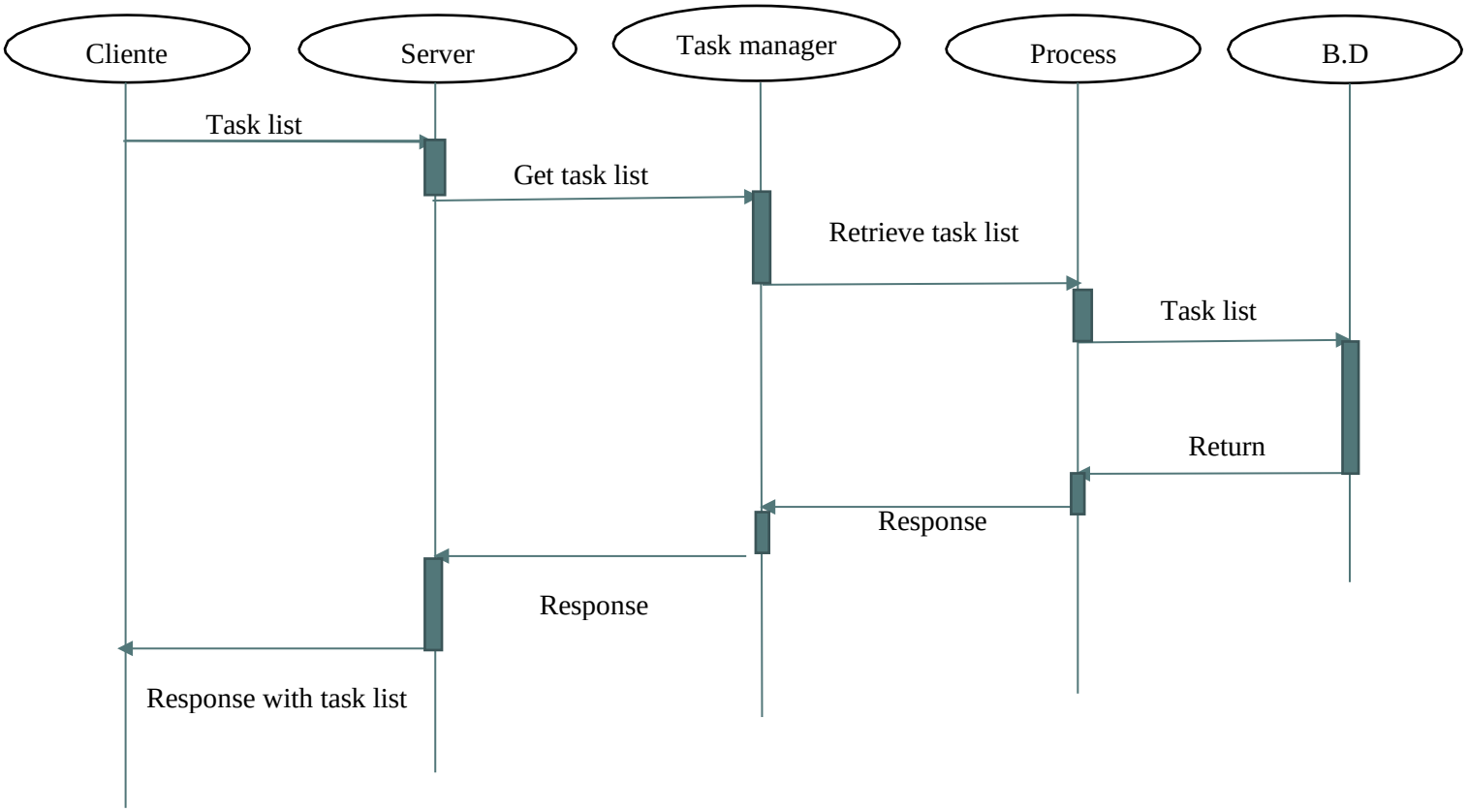
Crear tarea:



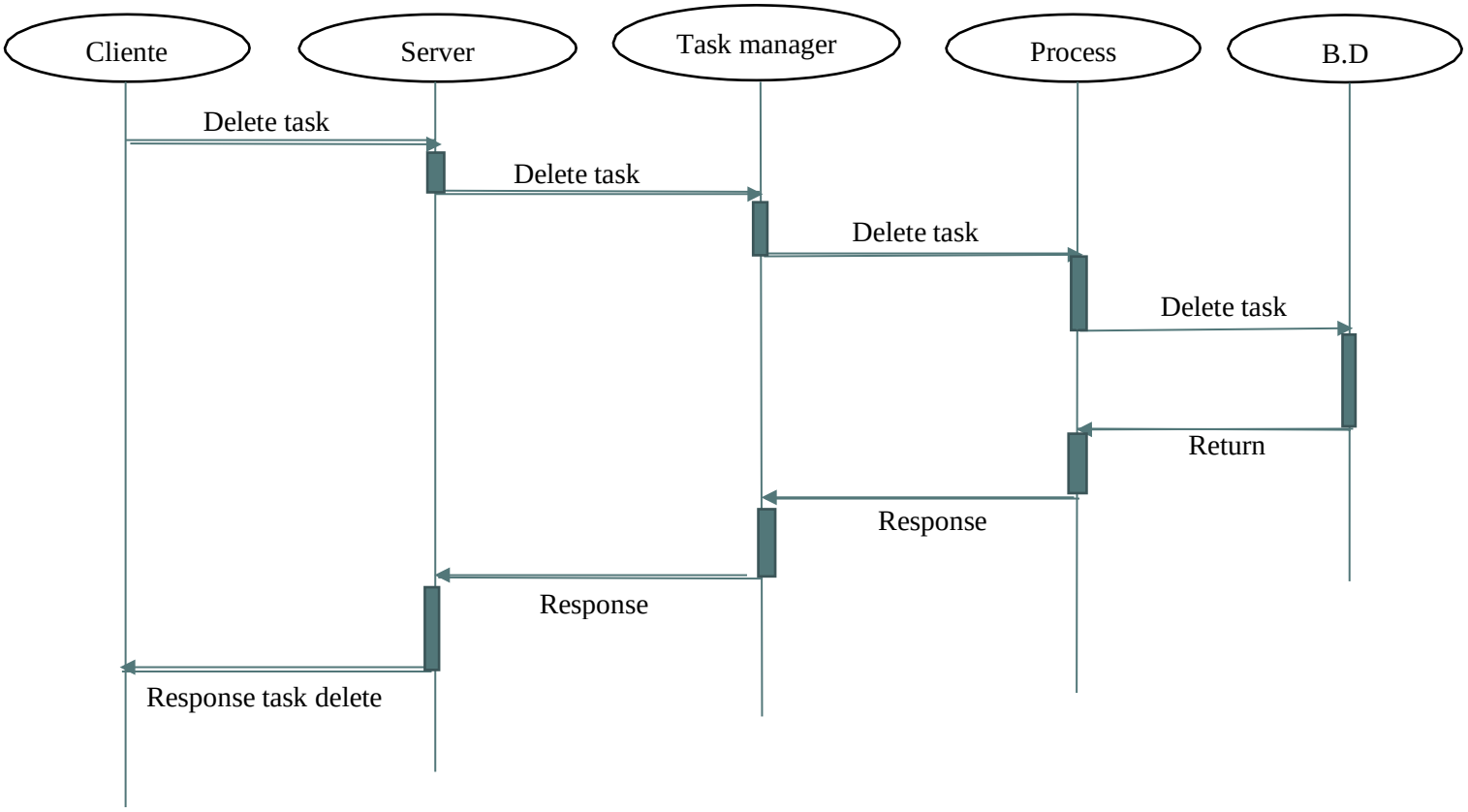
Editar tarea:



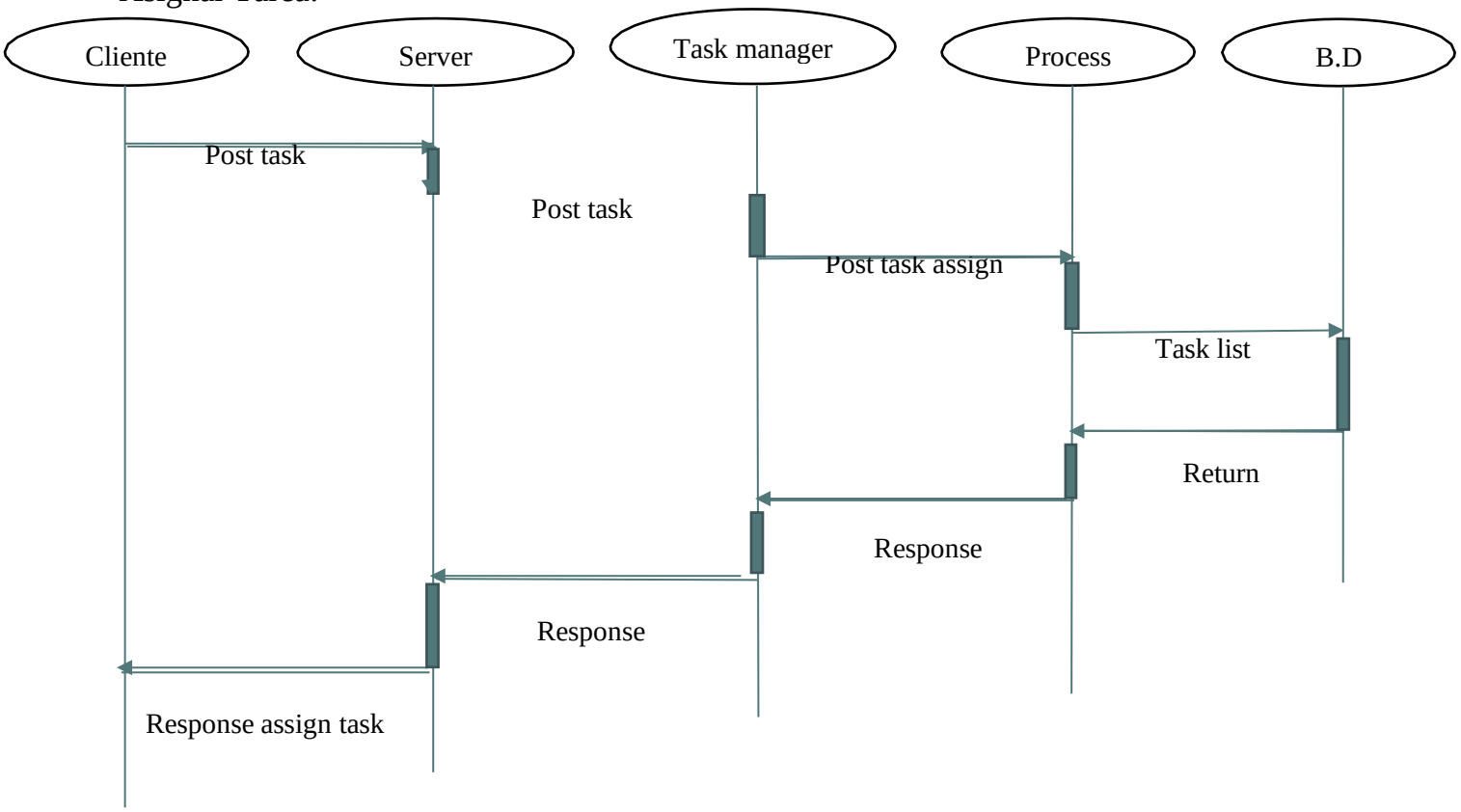
Listar tarea:



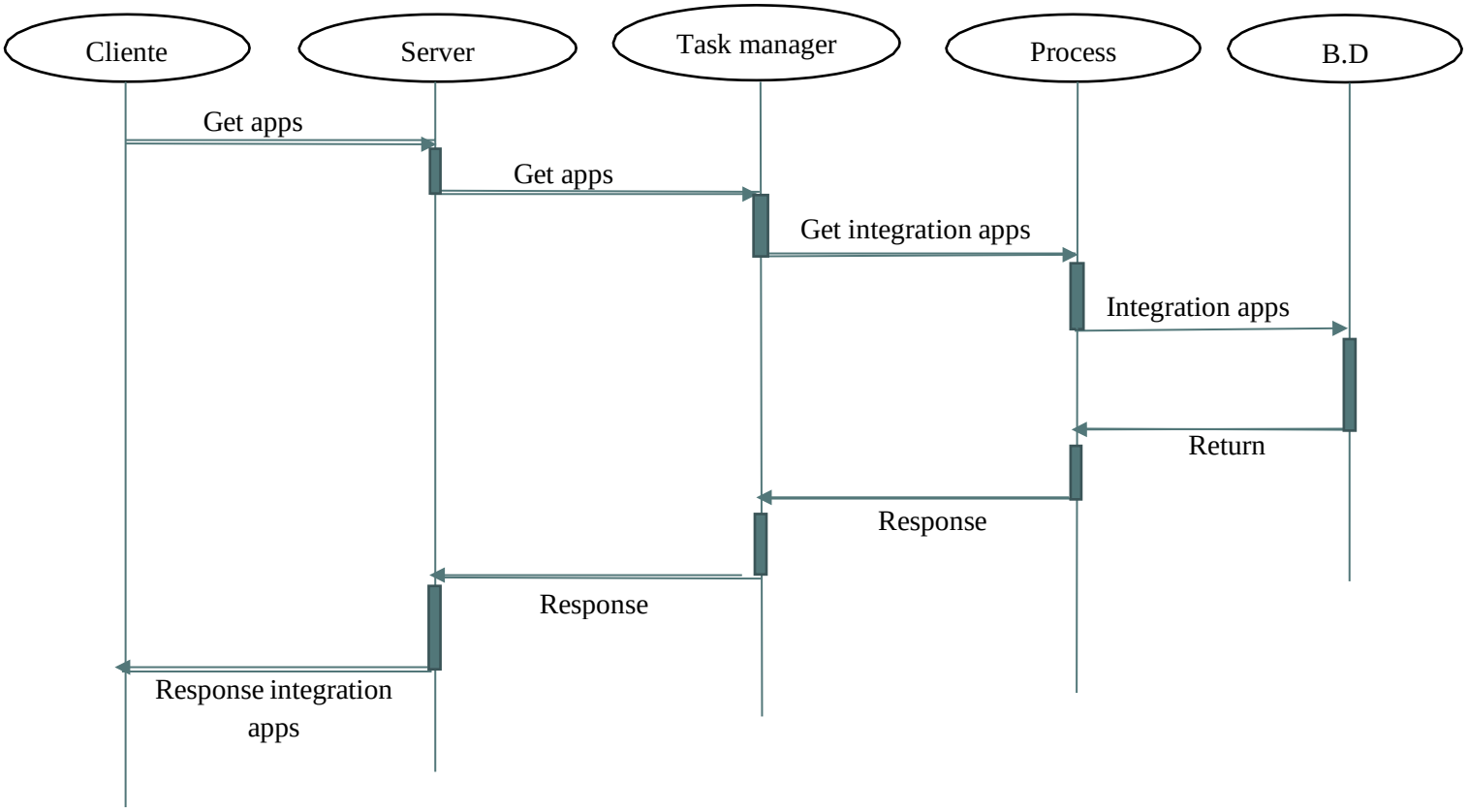
Eliminar tarea:



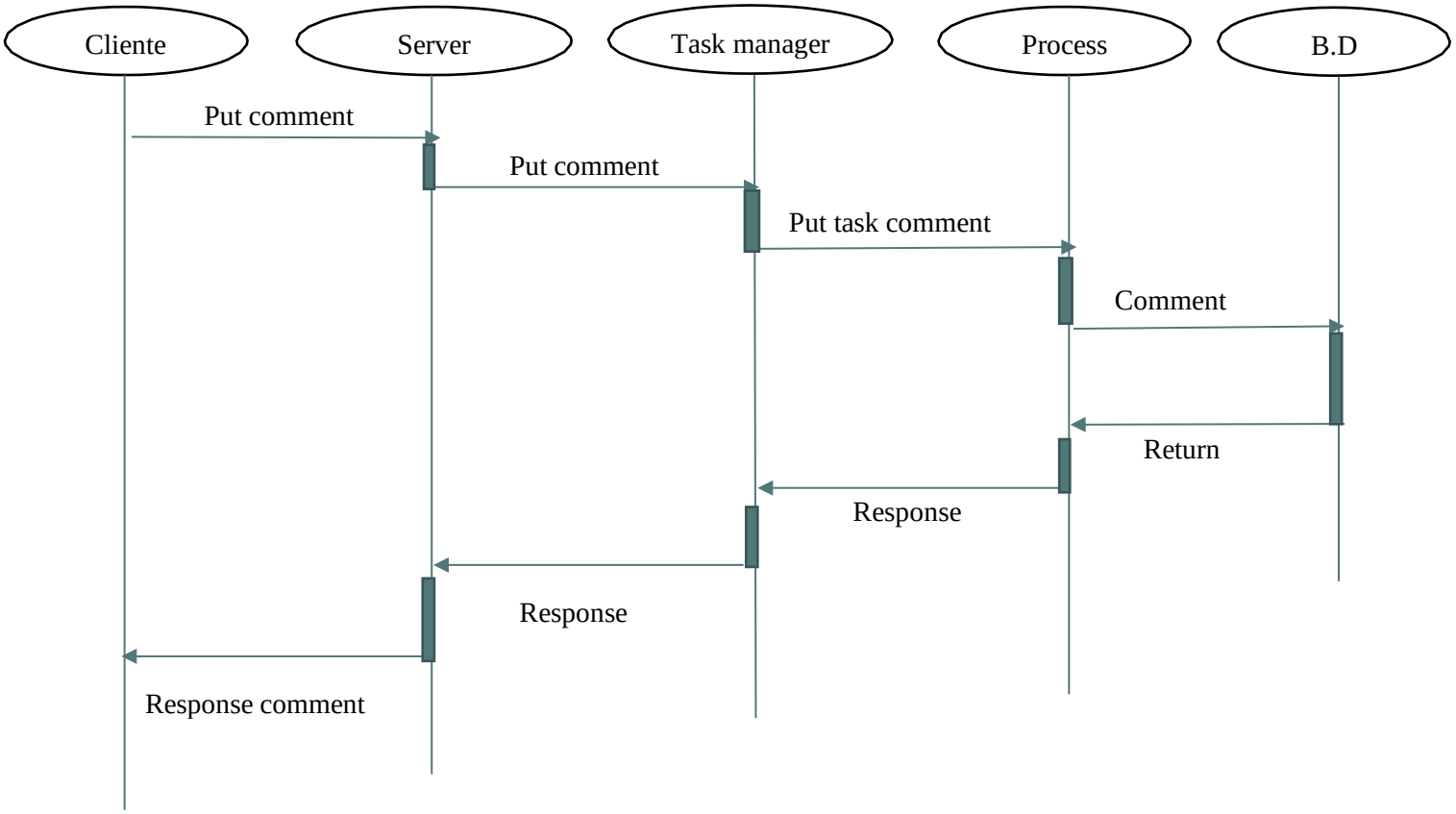
Asignar Tarea:



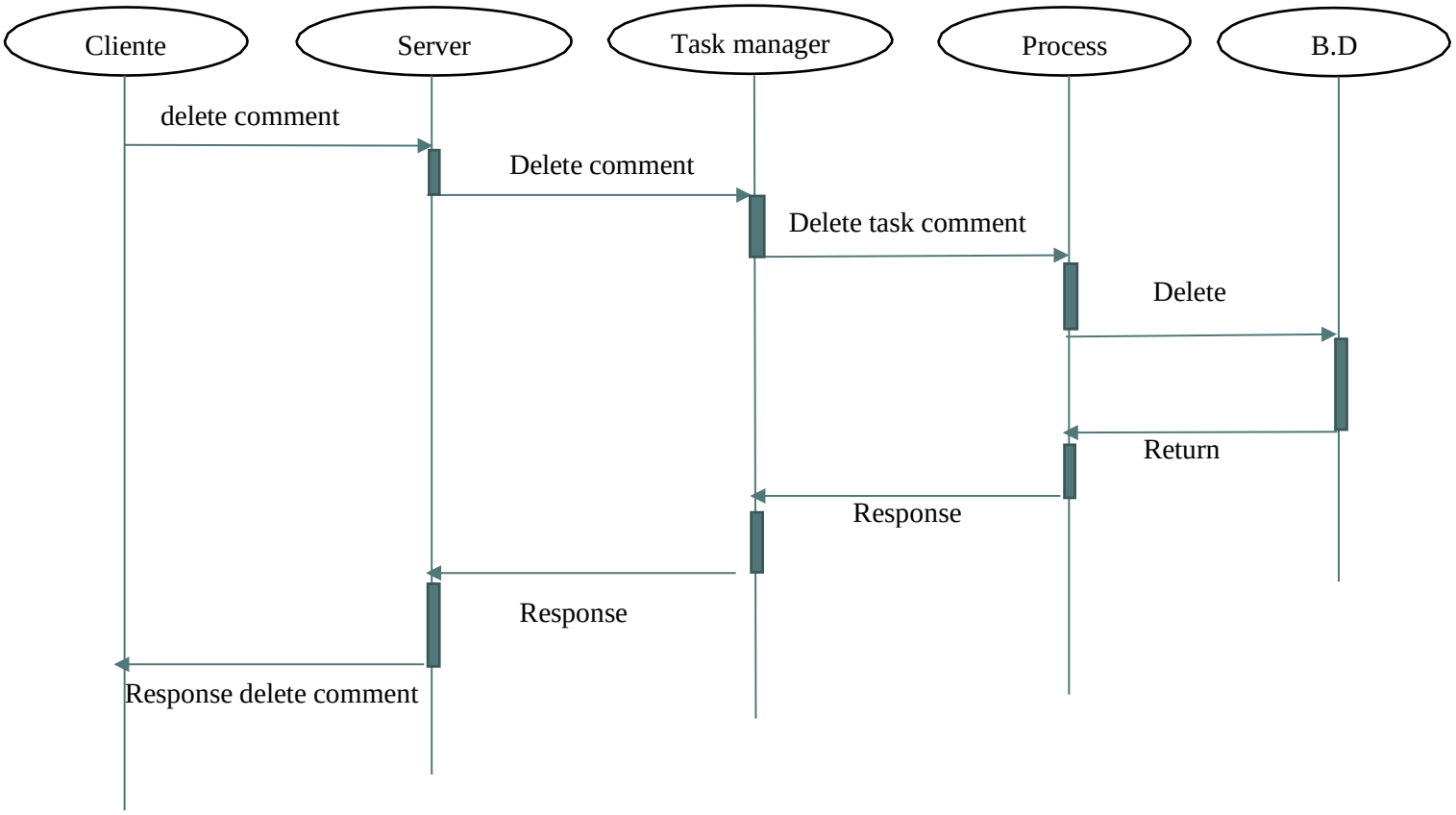
Integración con apps externas:



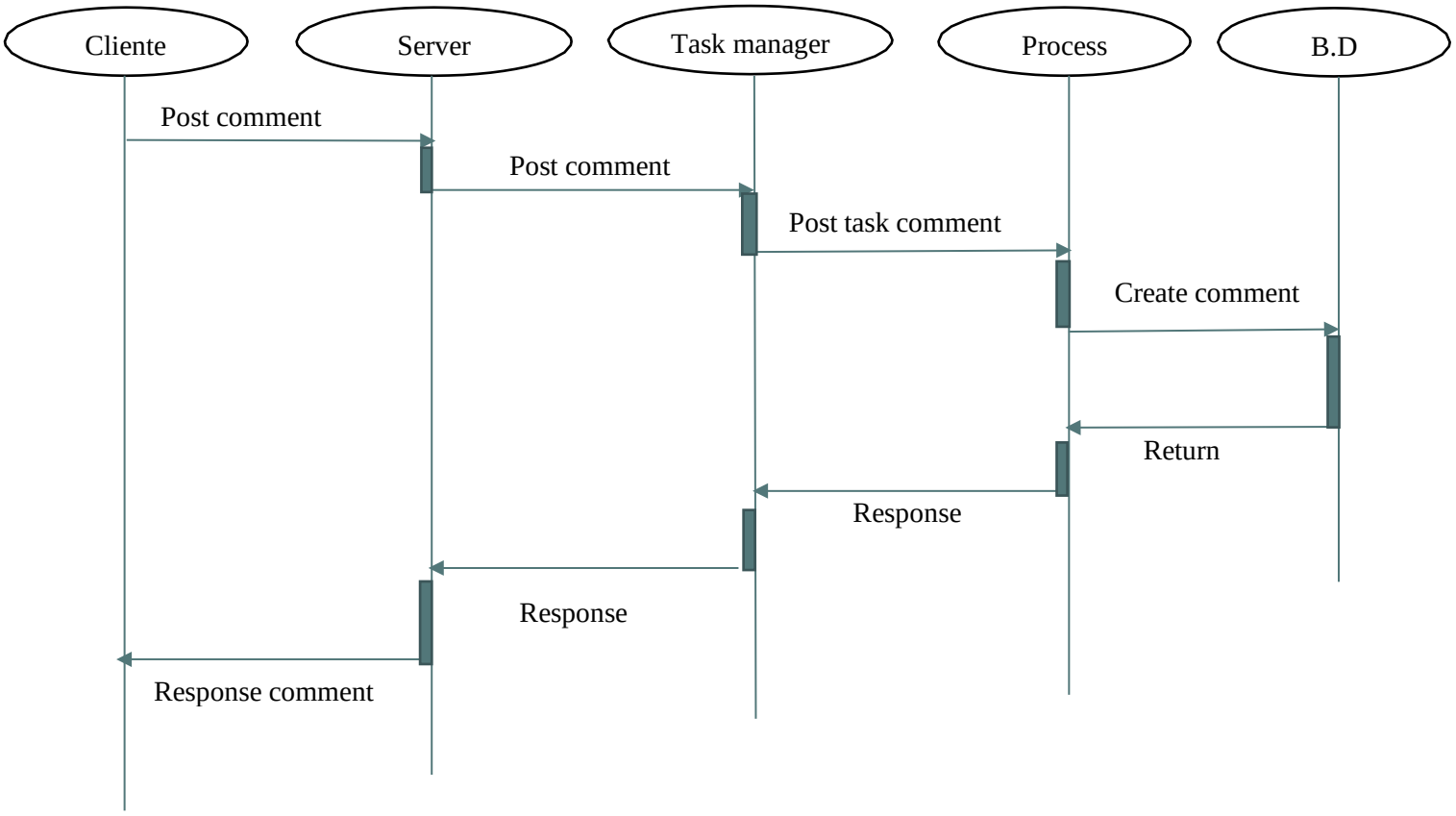
Comentar Tarea:



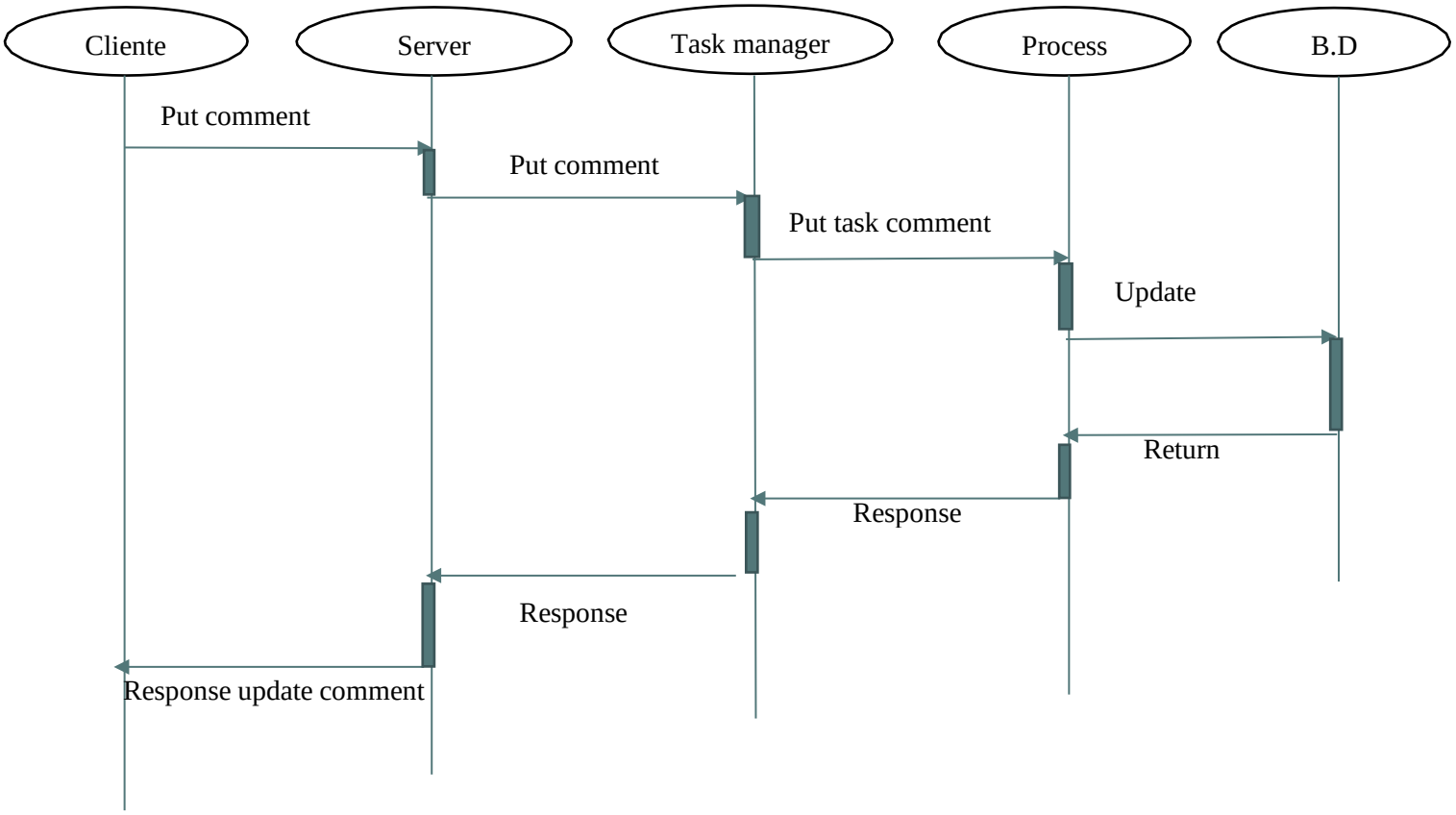
Borrar comentario tarea:



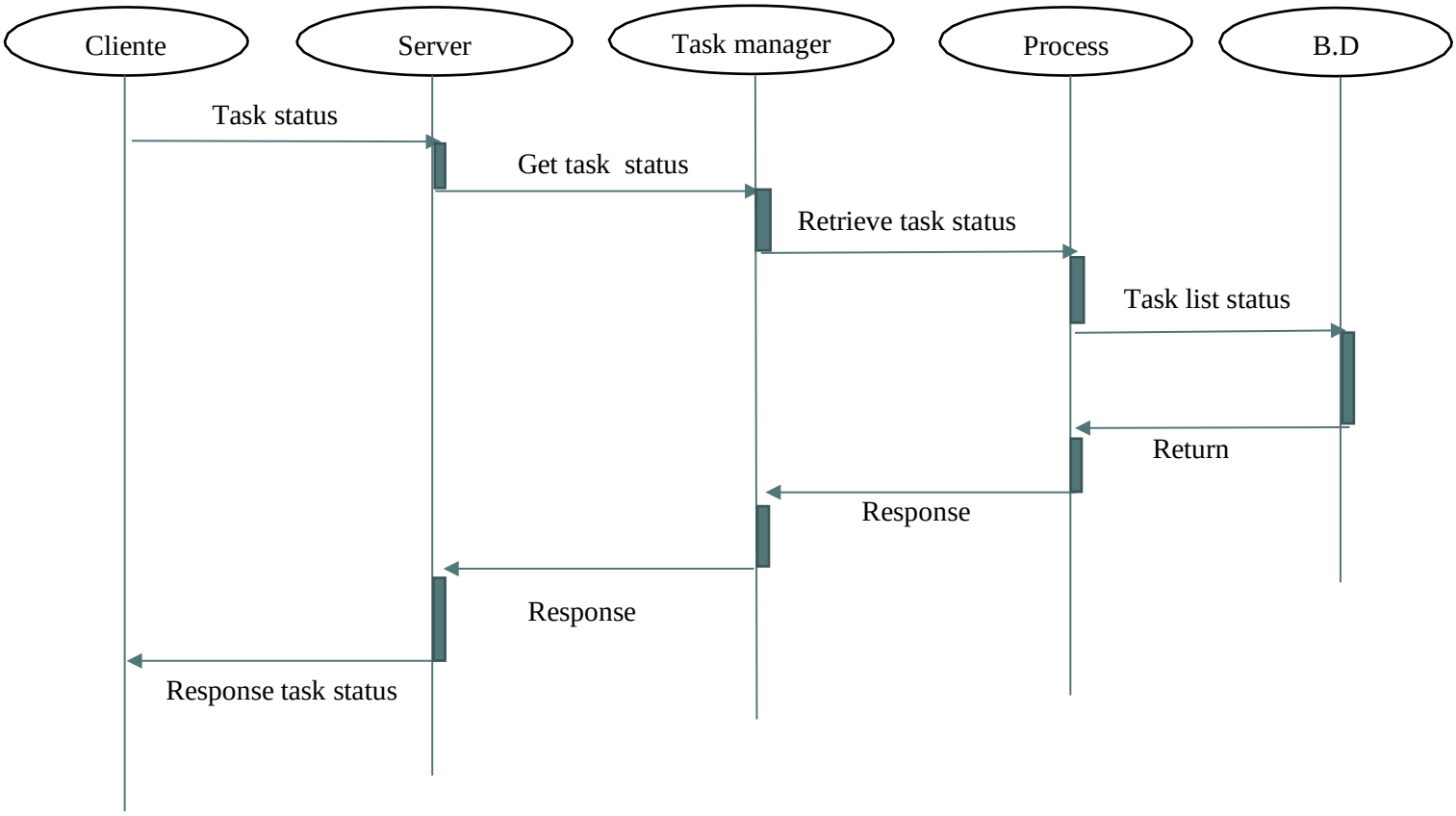
Responder comentario tarea:



Editar comentario tarea:



Estado Tarea:



Diagramas de Flujo de Casos de Uso

Nombre	**Crear Programa**
Actor Primario	Administrador del Sistema
Descripción	Este caso de uso describe el proceso que sigue un Administrador para crear un nuevo programa educativo dentro del sistema.
Precondiciones	• El Administrador debe estar autenticado en el sistema. • El Administrador debe tener permisos para crear programas.
Flujo de Eventos Principal	• El Administrador selecciona la opción "Crear Programa" en el menú de administración. • El sistema muestra la página "Crear Programa" con los campos necesarios para crear un nuevo programa. • El Administrador ingresa el nombre, la descripción y los objetivos del nuevo programa en los campos correspondientes. • El Administrador selecciona los cursos que formarán parte del programa. • El Administrador hace clic en el botón "Crear Programa". • El sistema valida la información ingresada. • Si la información es válida, el sistema crea el nuevo programa y muestra un mensaje de confirmación al Administrador. • Si la información no es válida, el sistema muestra un mensaje de error al Administrador y le solicita que corrija la información ingresada.
Flujo de Eventos Alternativos	• Editar Programa • Eliminar Programa • Asignar Cursos a un Programa
Postcondiciones	Se ha creado un nuevo programa en el sistema con los cursos seleccionados por el Administrador.
Actores Involucrados	Administrador
Reglas Condicionadas al Programa	• El nombre del programa debe ser único y no debe contener caracteres especiales. • La descripción del programa debe tener un mínimo de 20 caracteres. • El programa debe tener al menos un curso asignado.
Suposiciones Y Problemas	• Se asume que el Administrador tiene los conocimientos necesarios para crear y asignar cursos a un programa. • Un problema potencial es que se puedan crear programas con cursos que no estén alineados con los objetivos educativos de la institución.
Verificación	Programa Creado

Nombre	**Crear Curso**
Actor Primario	Administrador del Sistema
Descripción	Este caso de uso describe el proceso que sigue un Administrador para crear un nuevo curso dentro del sistema.
Precondiciones	• El Administrador debe estar autenticado en el sistema. • El Administrador debe tener permisos para crear cursos.
Flujo de Eventos Principal	• El Administrador selecciona la opción "Crear Curso" en el menú de administración. • El sistema muestra la página "Crear Curso" con los campos necesarios para crear un nuevo curso. • El Administrador ingresa el nombre, la descripción y el contenido del nuevo curso en los campos correspondientes. • El Administrador selecciona los instructores que impartirán el curso. • El Administrador hace clic en el botón "Crear Curso". • El sistema valida la información ingresada. • Si la información es válida, el sistema crea el nuevo curso y muestra un mensaje de confirmación al Administrador. • Si la información no es válida, el sistema muestra un mensaje de error al Administrador y le solicita que corrija la información ingresada.
Flujo de Eventos Alternativos	• Editar Curso • Eliminar Curso • Asignar Curso a un Programa
Postcondiciones	Se ha creado un nuevo curso en el sistema con los instructores seleccionados por el Administrador.
Actores Involucrados	Administrador
Reglas Condicionadas al Curso	• El nombre del curso debe ser único y no debe contener caracteres especiales. • La descripción del curso debe tener un mínimo de 20 caracteres. • El curso debe tener al menos un instructor asignado.
Suposiciones y Problemas	• Se asume que el Administrador tiene los conocimientos necesarios para crear y asignar instructores a un curso.
Verificación	Curso Creado

Nombre	**Crear Usuario**
Actor Primario	Administrador del Sistema
Descripción	Este caso de uso describe el proceso que sigue un Administrador para crear un nuevo usuario dentro del sistema.
Precondiciones	• El Administrador debe estar autenticado en el sistema. • El Administrador debe tener permisos para crear usuarios.
Flujo de Eventos Principal	• El Administrador selecciona la opción "Crear Usuario" en el menú de administración. • El sistema muestra la página "Crear Usuario" con los campos necesarios para crear un nuevo usuario. • El Administrador ingresa el nombre, el correo electrónico y la contraseña del nuevo usuario en los campos correspondientes. • El Administrador asigna un rol al nuevo usuario. • El Administrador hace clic en el botón "Crear Usuario". • El sistema valida la información ingresada. • Si la información es válida, el sistema crea el nuevo usuario y muestra un mensaje de confirmación al Administrador. • Si la información no es válida, el sistema muestra un mensaje de error al Administrador y le solicita que corrija la información ingresada.
Flujo de Eventos Alternativos	• Editar Usuario • Eliminar Usuario • Asignar Rol a un Usuario
Postcondiciones	Se ha creado un nuevo usuario en el sistema con el rol asignado por el Administrador.
Actores Involucrados	Administrador
Reglas Condicionadas al Usuario	• El correo electrónico del usuario debe ser único. • La contraseña debe cumplir con los requisitos de seguridad establecidos. • El usuario debe tener un rol asignado.
Suposiciones y Problemas	• Se asume que el Administrador tiene los conocimientos necesarios para crear y asignar roles a un usuario.
	• Un problema potencial es que se puedan crear usuarios con roles que no estén alineados con las políticas de seguridad de la organización.
Verificación	Usuario Creado

Nombre	Crear Institución
Actor Primario	Administrador del Sistema
Descripción	Este caso de uso describe el proceso que sigue un Administrador para crear una nueva institución dentro del sistema.
Precondiciones	• El Administrador debe estar autenticado en el sistema. • El Administrador debe tener permisos para crear instituciones.
Flujo de Eventos Principal	• El Administrador selecciona la opción "Crear Institución" en el menú de administración. • El sistema muestra la página "Crear Institución" con los campos necesarios para crear una nueva institución. • El Administrador ingresa el nombre, la dirección y los detalles de contacto de la nueva institución en los campos correspondientes. • El Administrador hace clic en el botón "Crear Institución". • El sistema valida la información ingresada. • Si la información es válida, el sistema crea la nueva institución y muestra un mensaje de confirmación al Administrador. • Si la información no es válida, el sistema muestra un mensaje de error al Administrador y le solicita que corrija la información ingresada.
Flujo de Eventos Alternativos	• Editar Institución • Eliminar Institución
Postcondiciones	Se ha creado una nueva institución en el sistema con los detalles proporcionados por el Administrador.
Actores Involucrados	Administrador
Reglas Condicionadas a la institución	El nombre de la institución debe ser único. La dirección y los detalles de contacto deben ser válidos.
Suposiciones y Problemas	• Se asume que el Administrador tiene los conocimientos necesarios para crear y gestionar una institución. • Un problema potencial es que se puedan crear instituciones con detalles incorrectos o incompletos.
Verificación	Institución Creada

Nombre	**Crear Rol**
Actor Primario	Administrador del Sistema
Descripción	Este caso de uso describe el proceso que sigue un Administrador para crear un nuevo rol dentro del sistema.
Precondiciones	• El Administrador debe estar autenticado en el sistema. • El Administrador debe tener permisos para crear roles.

Flujo de Eventos Principal	● El Administrador selecciona la opción "Crear Rol" en el menú de administración. ● El sistema muestra la página "Crear Rol" con los campos necesarios para crear un nuevo rol. ● El Administrador ingresa el nombre y la descripción del nuevo rol en los campos correspondientes. ● El Administrador selecciona los permisos que tendrá el nuevo rol en el sistema. ● El Administrador hace clic en el botón "Crear Rol". ● El sistema valida la información ingresada. ● Si la información es válida, el sistema crea el nuevo rol y muestra un mensaje de confirmación al Administrador. ● Si la información no es válida, el sistema muestra un mensaje de error al Administrador y le solicita que corrija la información ingresada.
Flujo de Eventos Alternativos	● Editar Rol ● Eliminar Rol ● Asignar Rol a un Usuario
Postcondiciones	Se ha creado un nuevo rol en el sistema con los permisos seleccionados por el Administrador.
Actores Involucrados	Administrador
Reglas Condicionadas al Rol	● El nombre del rol debe ser único y no debe contener caracteres especiales. ● La descripción del rol debe tener un mínimo de 10 caracteres. ● El rol debe tener al menos un permiso asignado.
Suposiciones y Problemas	● Se asume que el Administrador tiene los conocimientos necesarios para crear y asignar permisos a un rol. ● Un problema potencial es que se puedan crear roles con permisos que no estén alineados con las políticas de seguridad de la organización.
Verificación	Rol Creado

Descripción detallada de cada caso de uso:

1. **descripción detallada de casos de uso**		
2. **Descripción del caso de uso**		
Permite a los docentes en formación crear un plan de clase siguiendo las indicaciones y utilizando la plantilla proporcionada para tal fin.		
3. **Actor(es) : Docente en formación**		
4. **Precondiciones**		
● El docente en formación debe estar autenticado en el sistema.		
5. **Postcondiciones**		
● El plan de clase creado queda registrado en el sistema y está disponible para revisión por el docente asesor.		
6. **Flujo principal Pasos que describen la realización del caso de uso.**		
N.°	Acción del actor	Respuesta del sistema
1	Autenticación	El docente en formación se autentica en la plataforma con sus credenciales.
2	Acceder a la Sección de Planes de Clase	El docente en formación navega a la sección de planes de clase en la interfaz del software.
3	Crear Plan de Clase	El docente en formación selecciona la opción para crear un nuevo plan de clase.
4	Completar Plantilla	El docente en formación llena los campos de la plantilla: ● Área ● Asignatura ● Grado ● Grupo(s) ● Docente asesor ● Docente en formación ● Fecha o periodo de aplicación ● Duración (sesiones y horas) ● Tema ● Competencia ● Resultado de aprendizaje ● Contenidos ● Evidencia de aprendizaje ● Criterios de evaluación ● Actividades (Inicio, Desarrollo, Cierre) ● Mediación (Recursos)
5	Guardar Plan de Clase	El docente en formación guarda el plan de clase.
6	Notificación de Creación	El sistema envía una notificación al docente asesor indicando que hay un nuevo plan de clase para su revisión.

7	Fin	El caso de uso finaliza.
7. **Flujo alternativo [Pasos que describen la realización del caso de uso alternativo]**		
N.°	Acción del actor	Respuesta del sistema
1	Guardar Borrador	● El docente en formación puede guardar un borrador del plan de clase y continuar editándolo más tarde.
2	Formato no válido	● Si el sistema no admite un tipo de formato determinado, lo mostrará con un mensaje de error.
8. **Requisito Asociado (funcional, no funcional)**		

Requisitos funcionales:

1. El sistema debe permitir crear y guardar planes de clase usando la plantilla proporcionada.
2. El sistema debe enviar notificaciones de creación al docente asesor.

Requisitos NO funcionales:

1. La interfaz de creación de planes de clase debe ser intuitiva y fácil de usar.
2. Solo los usuarios autenticados deben poder crear y editar planes de clase.

Prioridad de Requerimientos

A partir del análisis de requerimientos, funcionalidades y el proceso de design thinking, se concreta la siguiente matriz de prioridad de requerimientos.

Para la interpretación se tiene en cuenta la siguiente escala con sus valores.

Eje de Urgencia:
- Obligatoria (5)
- Alta (4)
- Moderada (3)
- Menor (2)
- Baja (1)

Eje de Esfuerzo:
- Muy alto (5)
- Alto (4)
- Medio (3)
- Bajo (2)
- Muy bajo (1)

Esfuerzo	Urgencia				
	1-Baja	2-Menor	3-Moderada	4-Alta	5-Obligatorio
5-Muy alto	5	10 CU-6	15	20	25 CU-5
4-Alto	4	8	12 CU-3	16	20 CU-1 CU-2
3-Medio	3 CU-7 CU-9 CU-10	6 CU-4 CU-8	9	12	15
2-Bajo	2	4	6	8	10
1-Muy bajo	1	2	3	4	5

Requisitos No Funcionales

Requisitos de Desempeño

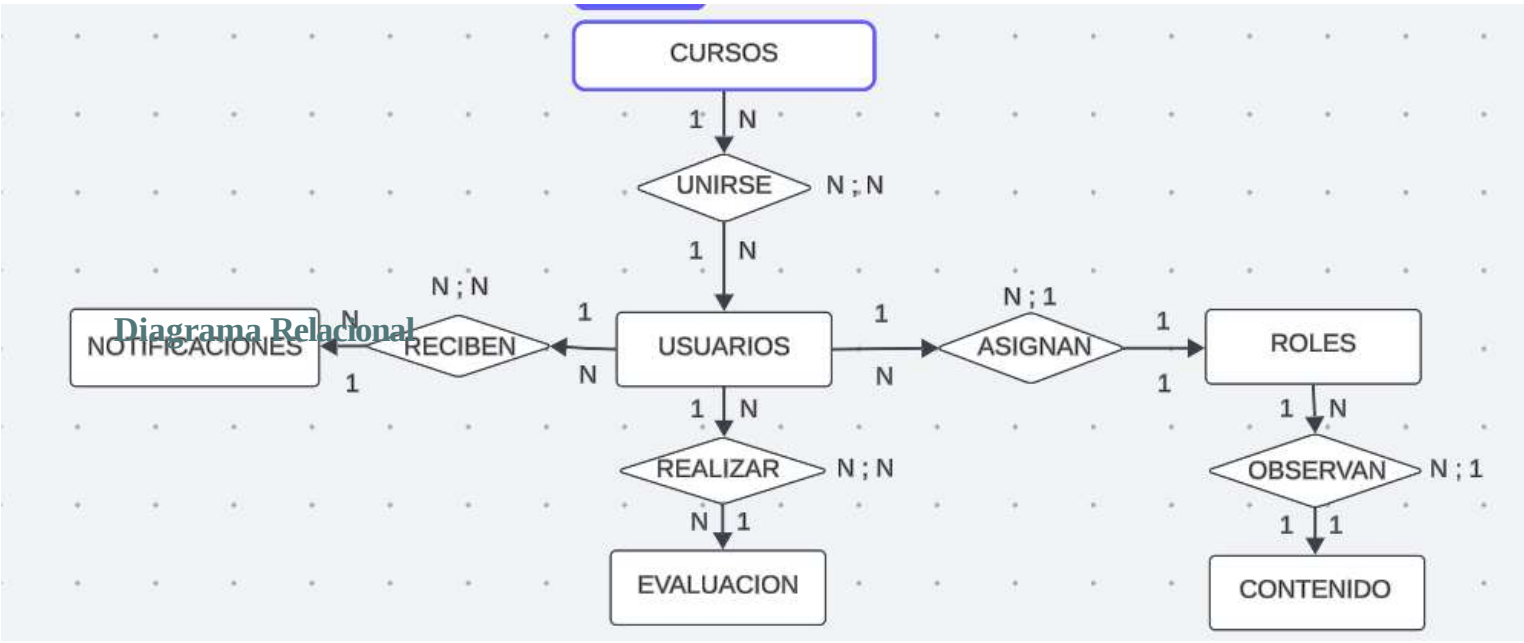
Requisitos de Seguridad

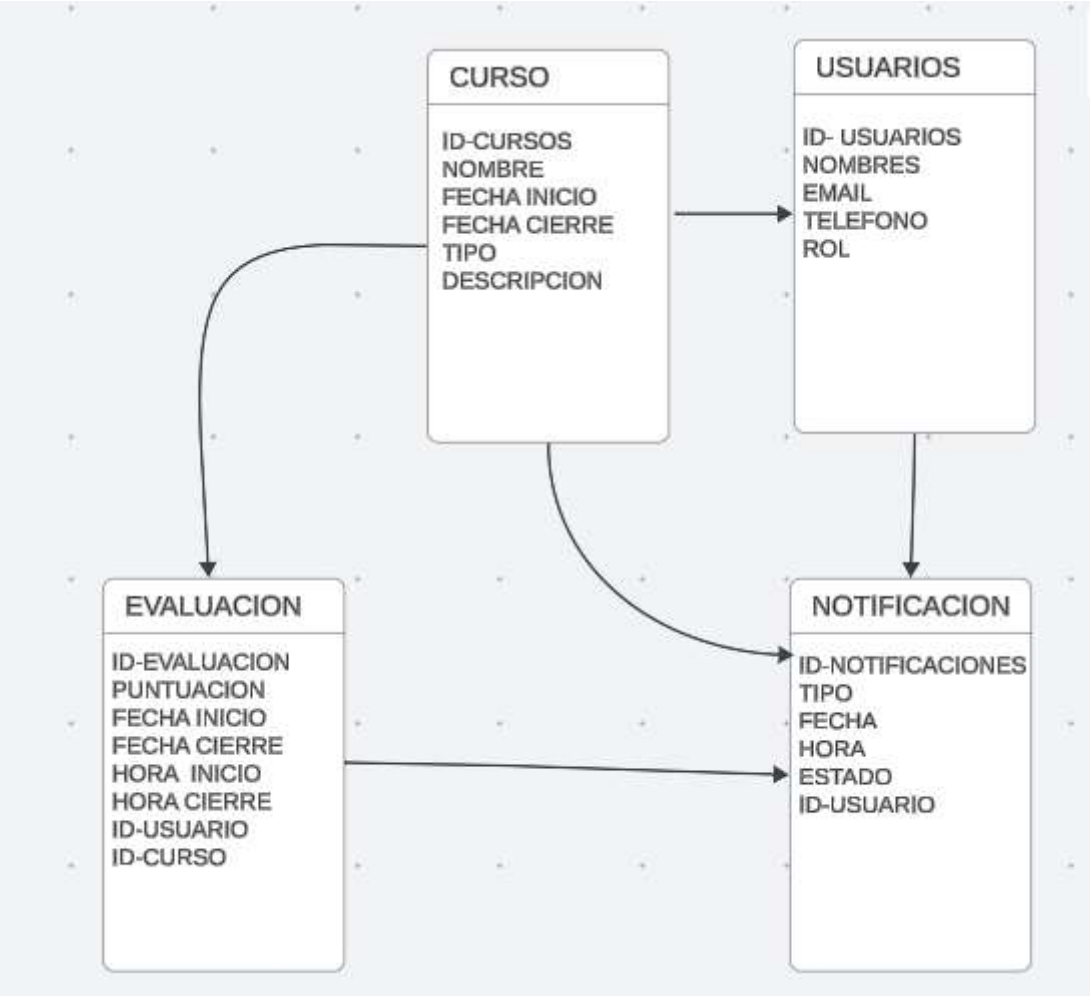
Requisitos de Usabilidad

Requisitos de Escalabilidad

Modelado E/R

Diagrama Relacional




CURSO
ID-CURSOS
NOMBRE
FECHA INICIO
FECHA CIERRE
TIPO
DESCRIPCION
USUARIOS
ID- USUARIOS
NOMBRES
EMAIL
TELEFONO
ROL
EVALUACION
ID-EVALUACION
PUNTUACION
FECHA INICIO
FECHA CIERRE
HORA INICIO
HORA CIERRE
ID-USUARIO
ID-CURSO
NOTIFICACION
ID-NOTIFICACIONES
TIPO
FECHA
HORA
ESTADO
ID-USUARIO

Script de modelo relacional

Curso {

ID: number,
Titulo: string,
Nombre: string,
Descripción: string,
Fecha – creación: timesdate,
Fecha – finalizacion: timesdate,
Estado: string,
}

Usuario{

ID: number,
Nombre: string,
Email: string
Telefono: numbers
Rol: string
}

Evaluación {

ID: number,
Punctuation: number
Fecha-creacion: timesdate,
Texto: string,
Estado: string,
ID_Usuario: array,
ID_Curso: array
}

Notificación {

ID: number,
Fecha-accion: timesdate ,
Tipo: string
Estado: string
Hora: timesdate
}

Descripción de Entidades y Relaciones

Reglas de Integridad

Anexos (si es necesario)

Diagramas Adicionales

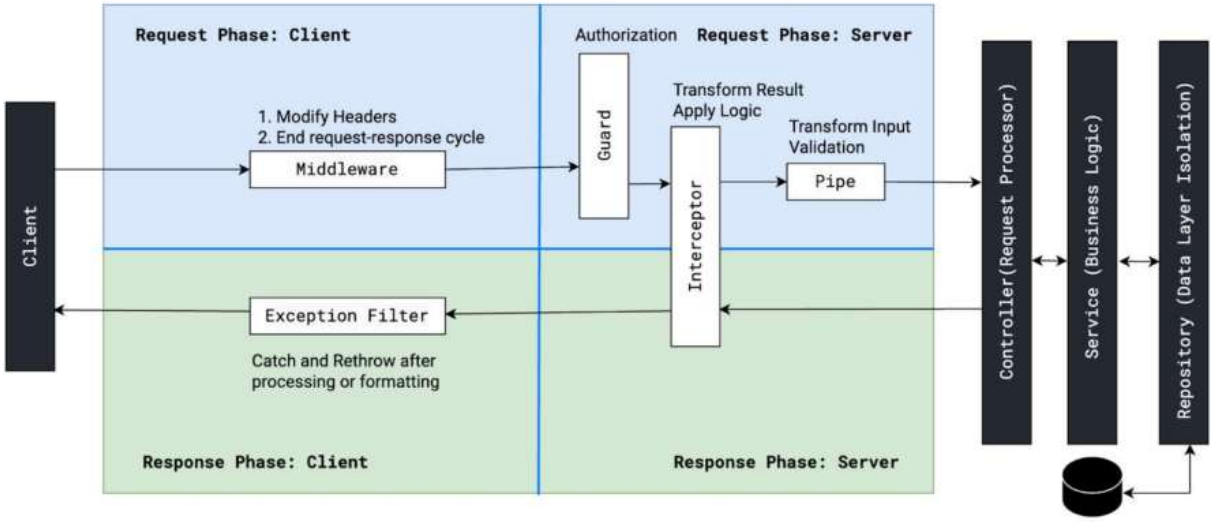
Referencias

Etapa 2: Persistencia de Datos con Backend

Descripción de la arquitectura propuesta

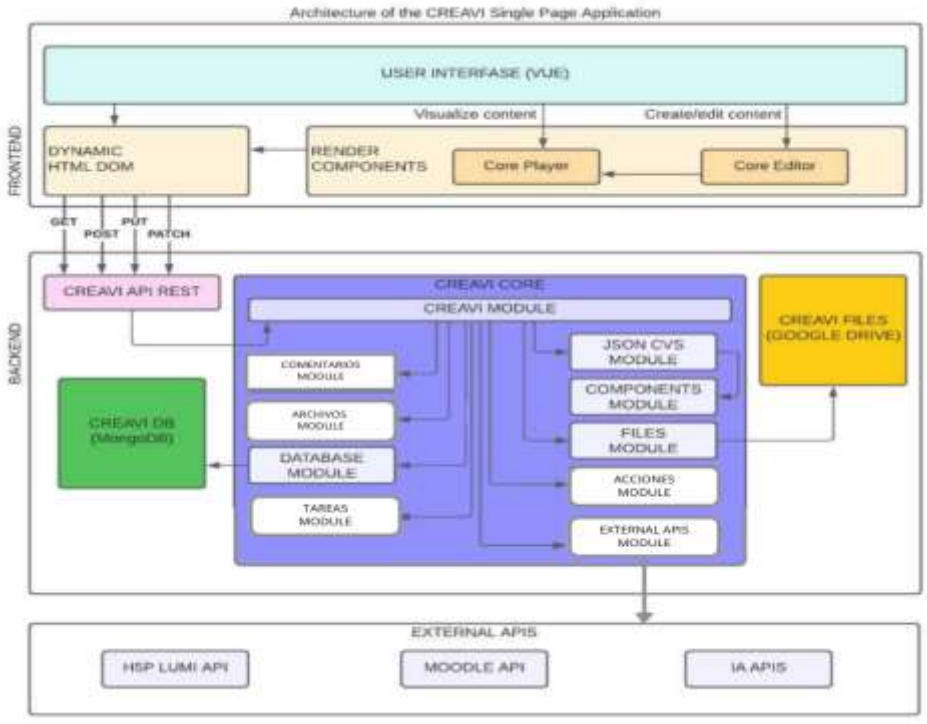
Este proyecto está basado en un sistema de Nest.js que cuenta con múltiples elementos, incluyendo el cliente, el interceptor que proporciona una validación al controlador, y una interacción fluida entre el controlador, el servidor y el repositorio, así como con la base de datos. El diseño de la arquitectura permite que cada capa del sistema se comunique de manera eficiente, asegurando que los datos se procesen correctamente antes de ser almacenados o enviados al usuario.

Diseño de la Arquitectura de Backend



Todo este proceso asegura que las solicitudes del cliente sean manejadas de manera ordenada y segura, facilitando la comunicación entre el cliente y el servidor. Cada componente tiene su rol específico para garantizar un funcionamiento eficiente y seguro de la aplicación.

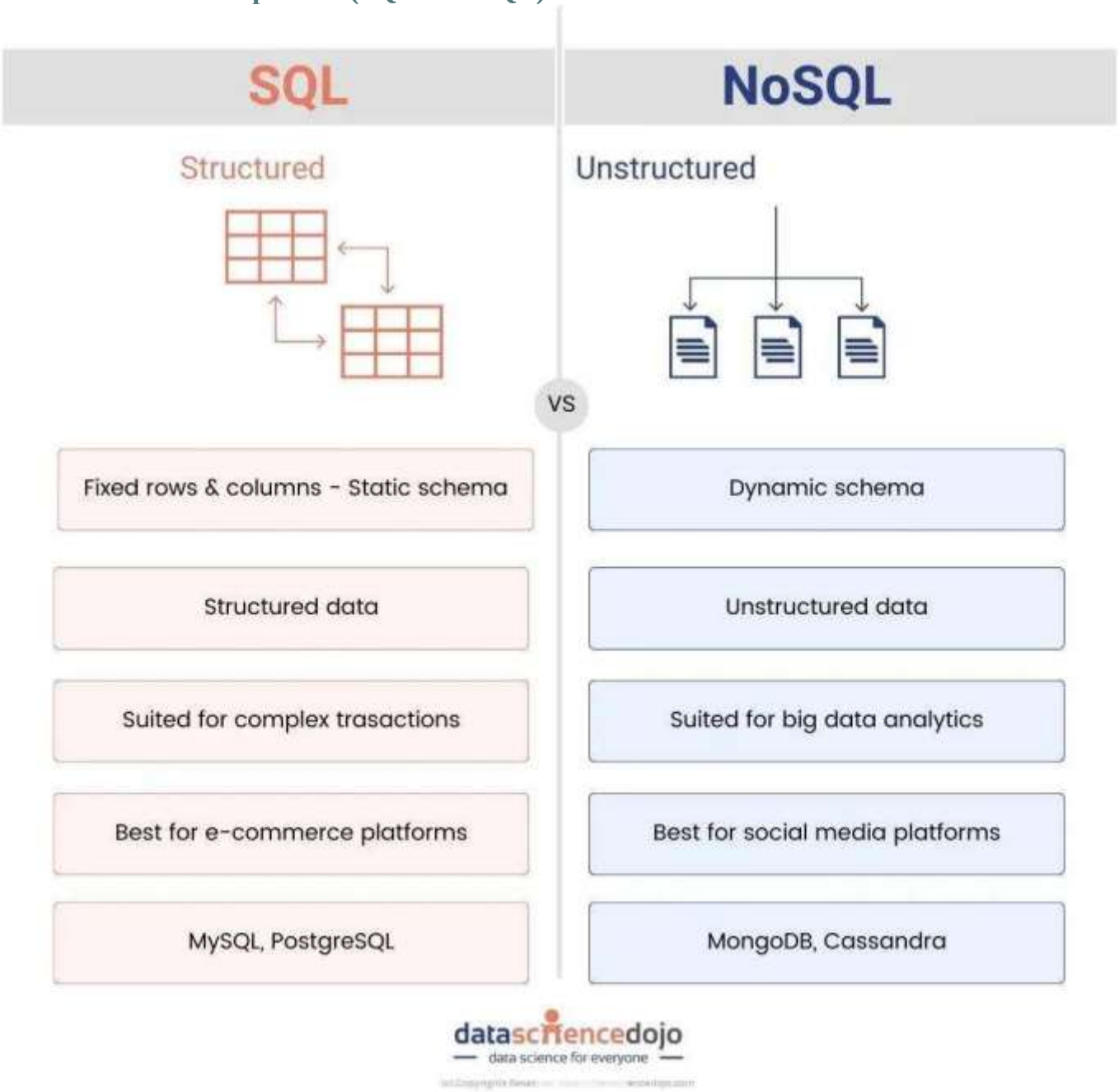
Diagramas de Arquitectura



Elección de la Base de Datos

Una de las decisiones más importantes para el desarrollo del backend de un proyecto es la elección de la base de datos. Entre las opciones más populares se encuentran SQL y NoSQL, y la selección dependerá de las necesidades específicas del proyecto.

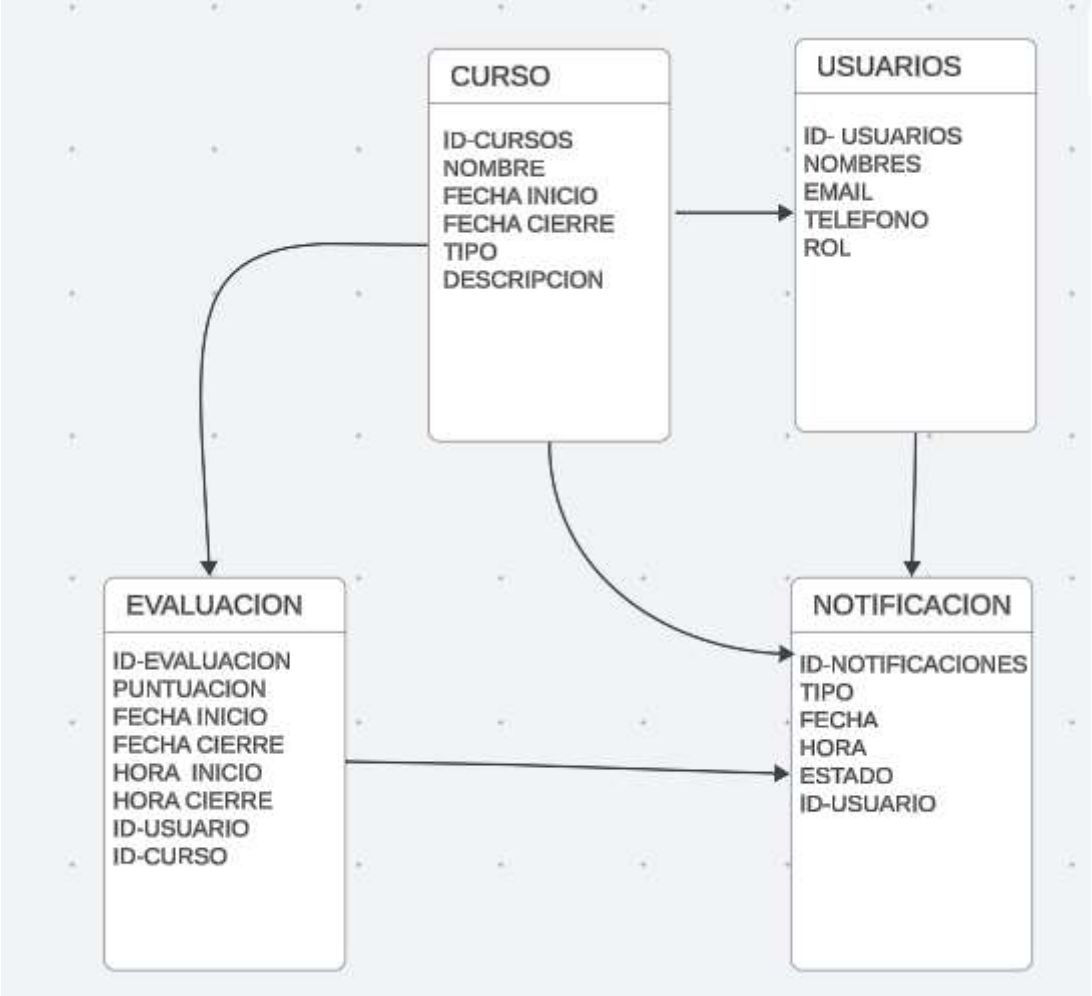
Evaluación de Opciones (SQL o NoSQL)



Justificación de la Elección

Para el desarrollo de este gestor de tareas se utilizará SQL debido a su capacidad para manejar una estructura más organizada. En este caso, las colecciones como tareas, archivos, acciones y comentarios requieren mantener unas conexiones claras entre ellas. SQL permite definir y gestionar estas relaciones con facilidad, asegurando consistencia en las operaciones del sistema, además de facilitar consultas complejas para el seguimiento y organización de la información.

Diseño de esquemas de bases de datos



El componente gestor de tareas organiza toda su información en cuatro colecciones principales: Tareas, que contiene los datos básicos de una tarea como título, descripción, fechas, prioridad, estado y etiquetas; Comentarios, donde se almacenan las opiniones o anotaciones vinculadas a cada tarea, con su texto, estado y fecha de creación del comentario; Archivos, que son los documentos adjuntos a las tareas, incluyendo su nombre, formato, URL, peso y Acciones, que registran los eventos realizados en las tareas, como creación, edición o eliminación, con detalles sobre la fecha, tipo de acción, motivo y estado posterior. Estas son las colecciones con las que cuenta el gestor de tareas y van a permitir que el usuario se le pueda garantizar una buena experiencia. Seguidamente, También encontramos una relación entre Tareas y Archivos, donde cada tarea puede tener múltiples archivos asociados que contienen documentos o recursos relacionados con la actividad, como imágenes, PDF o documentos de texto. También, existe una relación entre Comentarios y Acciones, ya que cada comentario puede generar una o varias acciones registradas en el sistema, como agregar, editar o eliminar, permitiendo mantener una vista clara sobre lo que se realiza dentro del gestor de tareas

Implementación del Backend

Elección del lenguaje de programación

Para el desarrollo del backend se eligió TypeScript como lenguaje de programación, aprovechando la estructura y la robustez que ofrece en proyectos complejos. Esta elección está fundamentada en la capacidad de TypeScript para proporcionar tipado estático y un desarrollo más estructurado y escalable en aplicaciones de gran envergadura. TypeScript es compatible con Nestjs**,** el framework de Node.js seleccionado, que permite la construcción de aplicaciones modulares, fácilmente testeables y mantenibles. NestJS se destaca en el desarrollo de Apis rest y aplicaciones orientadas a microservicios, lo que facilita la integración y escalabilidad del sistema.

Conexión a la Base de Datos

La conexión a la base de datos MongoDB Atlas desde el componente se realiza mediante las credenciales de acceso, la URL de conexión, y una configuración específica para asegurar el sistema.

Al trabajar con MongoDB, es necesario instalar las dependencias correspondientes ejecutando:

npm install @nestjs/mongoose mongoose.

```
1  import { Module } from '@nestjs/common';
2  import { AppController } from './app.controller';
3  import { AppService } from './app.service';
4  import { EvaluacionesModule } from './evaluaciones/evaluaciones.module';
5  import { NotificacionesModule } from './notificaciones/notificaciones.module';
6  import { CursosModule } from './cursos/cursos.module';
7  import { UsuariosModule } from './usuarios/usuarios.module';
8
```

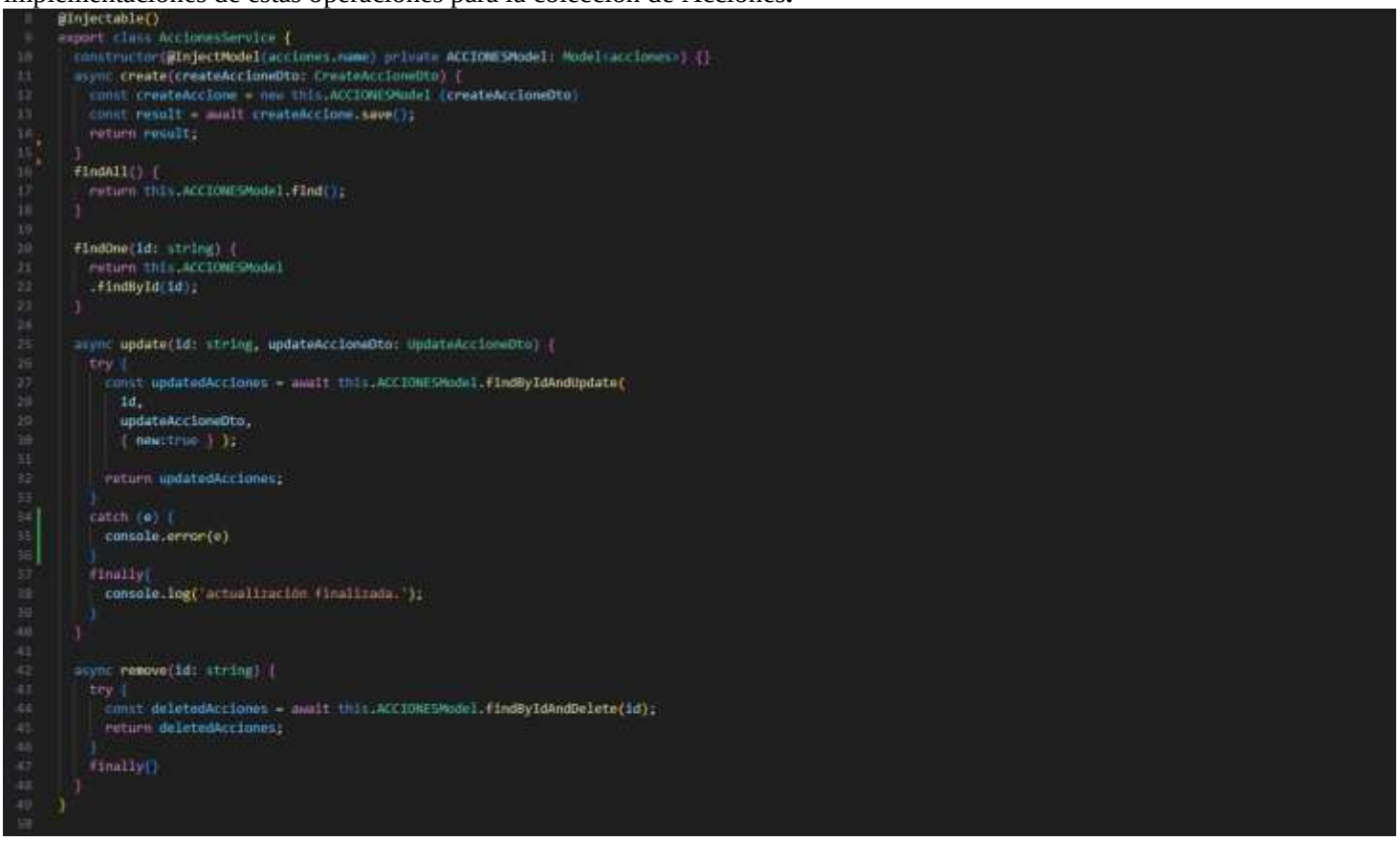
Luego, se realizó la configuración en el App Module. Aquí, el módulo MongooseModule se configuró para conectarse a la base de datos MongoDB usando la URI proporcionada, la cual incluye tanto el usuario y la contraseña como el nombre de la base de datos TaskManager_db. Esto permite el acceso seguro a la base de datos alojada en la nube.

```
10 @Module({
11   imports: [
12     MongooseModule.forRoot(
```

Por último, en el módulo principal (AppModule), se utiliza MongooseModule.forRoot() para establecer la conexión con MongoDB, especificando directamente la URI de conexión en el código.

Desarrollo de Operaciones CRUD

Las operaciones CRUD (Crear, Obtener todo, Obtener 1, Actualizar, Eliminar) son fundamentales para el funcionamiento óptimo del componente y permiten la interacción con las colecciones en MongoDB. A continuación, se detallan las implementaciones de estas operaciones para la colección de Acciones**.**



```
@Injectable()
export class AccionesService {
  constructor(@InjectModel(acciones.name) private ACCIONESModel: Model<acciones>) {}
  async create(createAccioneDto: CreateAccioneDto) {
    const createAccione = new this.ACCIONESModel (createAccioneDto)
    const result = await createAccione.save();
    return result;
  }
  findAll() {
    return this.ACCIONESModel.find();
  }

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

  async update(id: string, updateAccioneDto: UpdateAccioneDto) {
    try {
      const updatedAcciones = await this.ACCIONESModel.findByIdAndUpdate(
        id,
        updateAccioneDto,
        { new:true } );

      return updatedAcciones;
    }
    catch (e) {
      console.error(e)
    }
    finally{
      console.log('actualización finalizada.');
    }
  }

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

Cada una de estas funciones implementa una operación CRUD específica:

- **create**: Inserta un nuevo documento en la colección de Acciones utilizando un DTO (Data Transfer Object) para manejar los datos de entrada.
- **findAll**: Devuelve un conjunto de documentos predefinidos que simulan el listado completo de acciones.
- **findOne**: Devuelve un documento específico según el id proporcionado.
- **update**: Actualiza los campos de un documento existente con los valores proporcionados en el updateAccioneDto.
- **remove**: Elimina un documento basado en el id especificado

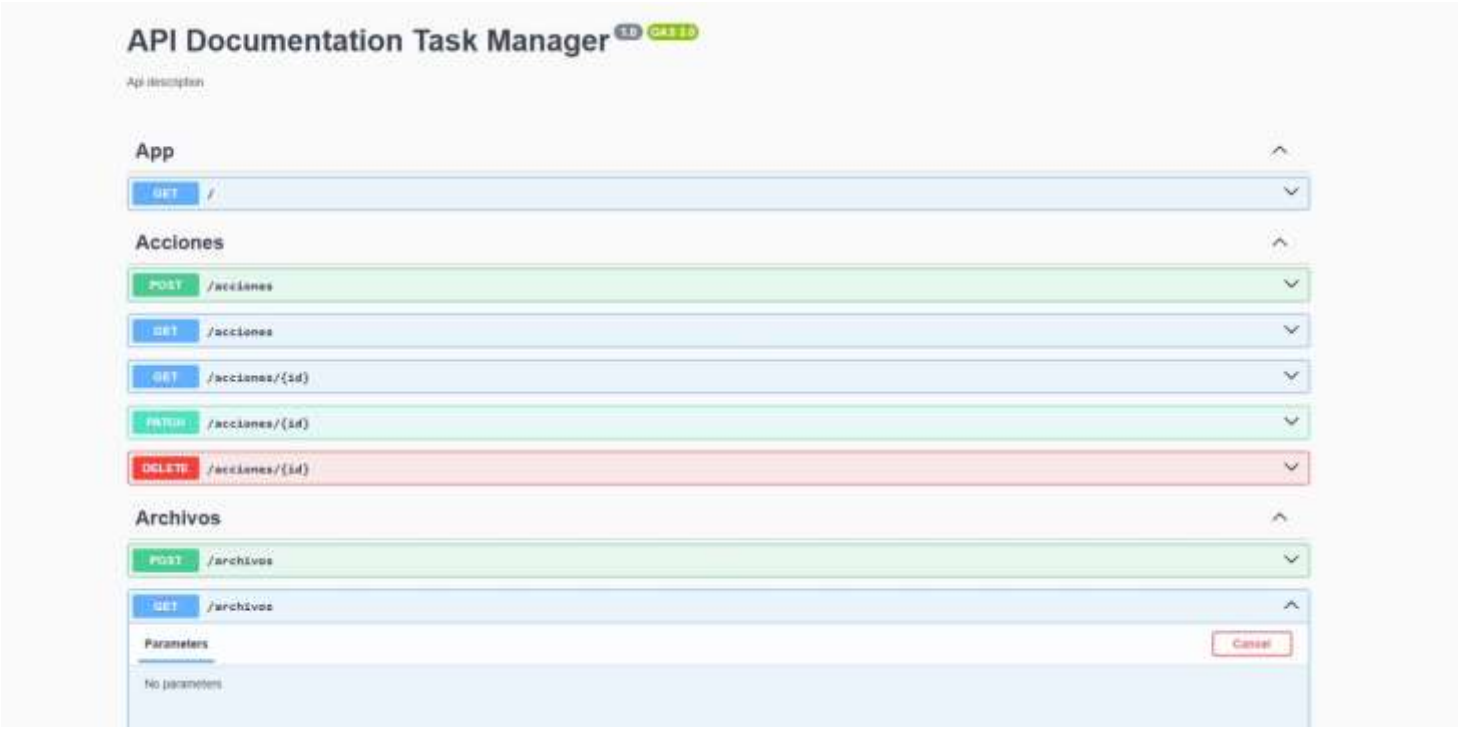
API DOCUMENTACIÓN TASK MANAGER.

Este código implementa una aplicación NestJS configurada para utilizar Swagger como herramienta de documentación interactiva para una API. Comienza importando los módulos esenciales: NestFactory para

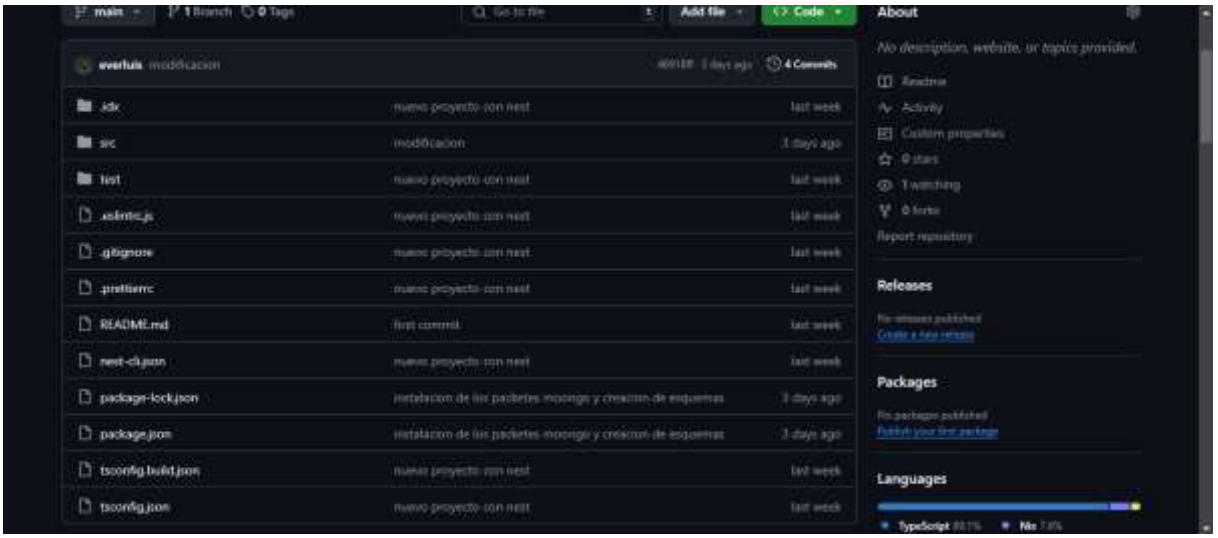
inicializar la aplicación, AppModule como módulo principal y los componentes de @nestjs/swagger para la configuración de Swagger.

En la función bootstrap(), se inicializa la aplicación mediante NestFactory.create(AppModule). A continuación, se configura la documentación con DocumentBuilder, definiendo título, descripción y versión de la API. Luego, se genera el documento Swagger con SwaggerModule.createDocument, que combina la configuración creada y la instancia de la aplicación. Finalmente, se expone la documentación Swagger en la ruta /api mediante SwaggerModule.setup y se inicia el servidor escuchando en el puerto especificado (process.env.PORT o 3000 por defecto) con app.listen. Este enfoque garantiza una API documentada y lista para su consumo.

```typescript
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { DocumentBuilder, SwaggerModule} from "@nestjs/swagger";

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

  const config = new DocumentBuilder()
    .setTitle('API Documentation Task Manager')
    .setDescription('Api description')
    .setVersion('1.0')
    .build();

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

  await app.listen(process.env.PORT ?? 3000);
}
bootstrap();
```



Enlace de conectividad asía el repositorio de github.



CREACION Y APLICACIÓN DE LOS ESQUEMAS (SCHEMAS)

```typescript
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { Document } from 'mongoose';

@Schema({ timestamps: true })
export class cursos  extends Document {

  @Prop()
  nombre: string;

  @Prop()
  fechainicio: string;

  @Prop()
  fechacierre: string;

  @Prop()
  tipo: string;

  @Prop()
  descripcion: string;

  @Prop({ default: () => new Date() })
  timestamp: Date;
}

export const cursosSchema = SchemaFactory.createForClass(cursos);
```

Etapa 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