UNIVERSIDAD DE CÓRDOBA

acreditada INSTITUCIONALMENTE

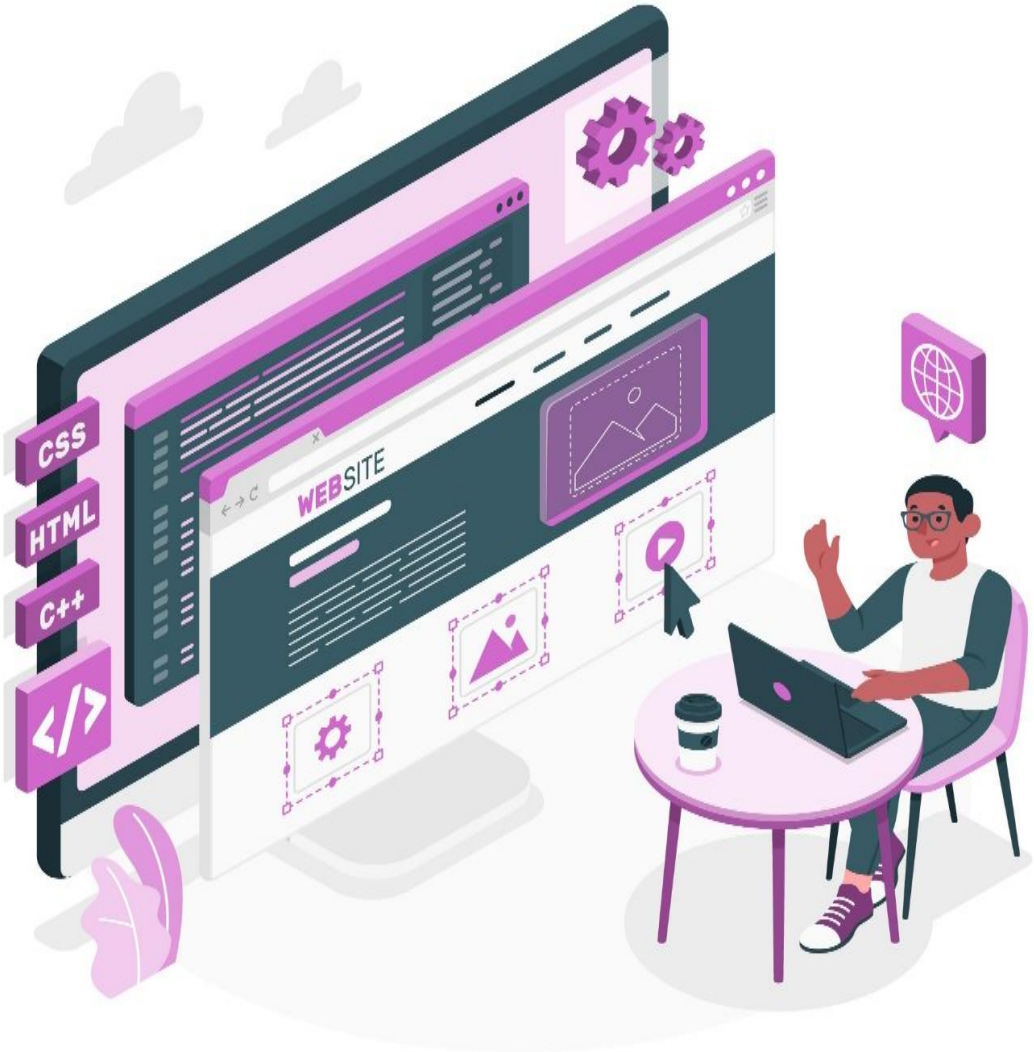
LICENCIATURA EN INFORMÁTICA Y MEDIOS AUDIOVISUALES

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

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

Módulo de Creación de Contenidos.

Andrea de los Ángeles Bettin Cáceres
Félix Mateo Roldan Rodríguez
Gregorio Pérez Valencia
Juan Daniel Pérez Argel



Grupo-Investigacion-Bimadino/paul_content_module (github.com)

Descripción del software

El módulo de contenidos es una parte fundamental en el desarrollo del software CREAVI, ya que se encarga de establecer los conceptos teóricos que sustentan todo el sistema. En esta sección, se definen los calificativos que describen los contenidos tratados en el software de manera general.

Dentro del módulo de contenidos, es importante incluir las definiciones conceptuales. Estas definiciones son la expresión de los rasgos fundamentales que constituyen el contenido de cada concepto. En esta fase, se presenta la declaración teórica de cada concepto, brindando una comprensión clara y precisa de su significado.

Además, es necesario especificar las características necesarias de cada definición conceptual. Esto implica puntualizar los atributos representativos de cada palabra extraída de las bases conceptuales. Estas características proporcionan coherencia y consistencia a los contenidos del software, asegurando que se mantenga una línea de pensamiento coherente en todo el desarrollo.

Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

1. Introducción

Propósito del Documento

El presente documento tiene como finalidad documentar el proceso de diseño del módulo de creación de contenido, dentro de la plataforma CREAVI. 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

- Etapa 2: Persistencia de Datos con Backend – Servidor

En la etapa 2, nos enfocamos en la creación del módulo "Creación de Contenidos" siguiendo los lineamientos establecidos en la etapa 1. Durante esta fase, se continúa desarrollando elementos de diseño e implementación de software específicos para la creación de contenidos en nuestro software educativo. Esto implica la integración de APIs, servidores o microservicios que respalden las aplicaciones cliente del software educativo en este módulo.

Para lograr esto, el curso se centrará en impartir conceptos relacionados con los sistemas de bases de datos, incluyendo el diseño lógico de las bases de datos, la organización de los sistemas gestores de bases de datos, y el uso de lenguajes de definición de datos y manipulación de datos en NoSQL.

Se abordarán temas clave, como el diseño y el modelo lógico y físico de las bases de datos, la gestión de sistemas de bases de datos. También se explorarán tecnologías cliente/servidor para la creación de servidores API, utilizando tecnologías modernas como node.js, express, Nest.js, Spring, entre otras.

Además, se definirán los componentes necesarios para acceder a estas bases de datos, incluida la creación del servidor API. Se utilizarán tecnologías de vanguardia, como servicios de hospedaje en la nube gratuitos, para desplegar la API.

- 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 Módulo Creación de contenidos

El "Alcance del Módulo Creación de Contenidos" tiene como objetivo proporcionar una estrategia sólida para optimizar la experiencia de docentes y alumnos al utilizar el software educativo. Esta estrategia está diseñada para abordar los desafíos comunes que surgen en la transmisión de información digital en el entorno educativo. Algunos de estos desafíos pueden incluir la necesidad de facilitar la comprensión de los contenidos, superar limitaciones de tiempo y recursos, y mejorar la interacción en el aula virtual.

Las funciones que caracterizan este módulo, tanto las que se encuentran disponibles, como las que estarán al servicio de los usuarios en un futuro están descritas a continuación.

- Creación de diferentes formatos de contenidos (textos, imágenes, videos, archivos).
- Permitir visualizar el contenido que se está creando.
- Crear categorías para organizar el contenido categóricamente.
- Permitir que otros usuarios puedan comentar y hacer observaciones a los contenidos.

A largo plazo se incorporarán nuevas funciones como:

- Permitir la colaboración de contenidos entre varios usuarios.
- Exportar los contenidos en los formatos que el usuario defina.

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).

2. Descripción General

Objetivos del Módulo

Proporcionar a docentes y usuarios del software educativo las herramientas y funcionalidades necesarias para la creación y gestión efectiva de materiales de aprendizaje, promoviendo la personalización, interactividad y calidad de la enseñanza, así como facilitando la innovación educativa.

Funcionalidad General

- **Creación de contenidos**: dentro de esta función el usuario podrá realizar acciones como crear, editar, pre visualizar y publicar el contenido.
- **Carga de Multimedia**: Esta permite a los usuarios cargar archivos multimediales como imágenes, videos, audios para acompañar los conceptos facilitando su entendimiento.
- **Inserción de hipervínculos**: Al momento de crear contenidos los usuarios podrán también insertar hipervínculos.
- **Categorizar el contenido**: Permite al usuario jerarquizar y categorizar los contenidos según el tema, formato o cualquier forma que permita categorizar el contenido.
- **Revisar y observar**: Los usuarios con esta función podrán revisar un contenido creado y hacerles observaciones, donde quien lo crea puede modificar y al final puede colocar un estado a esa observación, como puede ser: iniciado, en proceso, terminado.
- **Comentar**: Los usuarios pueden comentar sobre el contenido, algo que es muy diferente a la observación, cualquier usuario puede comentar más no todos los usuarios pueden revisar y hacer observaciones a los contenidos.
- **Pre visualizar**: Para llevar una idea de lo que se está creando y como va quedando el desarrollo los usuarios podrán tener una vista previa de ello al momento de estar creando.
- **Publicar**: Al finalizar la creación de un contenido, el usuario podrá publicar este para que esté disponible dentro de la plataforma.
- **Acceso móvil**: Permite a los usuarios acceder a la plataforma ver y crear contenido desde dispositivos móviles, como smartphones y tabletas, con una interfaz responsive para mejorar la accesibilidad y flexibilidad de uso.
- **Búsqueda y Filtros**: Facilita la búsqueda de contenidos en la plataforma, permitiendo a los usuarios encontrar materiales de manera efectiva mediante herramientas de filtrado que mejoran la organización y categorización de contenidos, optimizando la eficiencia de la plataforma.

Usuarios del Sistema

Los siguientes usuarios pueden interactuar con este módulo dependiendo de las funcionalidades.

Funcionalidad	Administradores	Docentes	Alumnos	Invitados
Creación de contenidos		✓	✓	
Carga de Multimedia		✓	✓	
Inserción de hipervínculos		✓	✓	
Categorizar el contenido		✓	✓	
Revisar y observar		✓	✓	
Comentar	✓	✓	✓	✓
Pre visualizar		✓	✓	
Publicar		✓	✓	
Acceso Móvil	✓	✓	✓	✓
Búsqueda y Filtros	✓	✓	✓	✓

Restricciones

Solo los usuarios con el rol docente pueden revisar un contenido para dar observaciones siempre y cuando lo requiera ya sea para modificar el contenido, el docente creador no está obligado a hacer el cambio y si lo hace o no tendrá un estado el cual puede ser: iniciado, en proceso o terminado.

3. Requisitos Funcionales

Creación de contenidos:

- Los usuarios deben poder crear contenido educativo de forma interactiva.
- Los usuarios deben tener acceso a herramientas de edición de texto.
- Deben poder guardar y editar sus contenidos en cualquier momento.

Carga de Multimedia:

- Los usuarios deben poder cargar y adjuntar archivos multimedia, como imágenes, videos y audio.
- Los usuarios deben tener acceso a herramientas de edición de imágenes y otros elementos multimedia.
- Se debe proporcionar una capacidad para organizar y gestionar los archivos multimedia cargados.

Inserción de hipervínculos:

- Los usuarios deben poder insertar hipervínculos en el contenido para enlazar a recursos externos, como sitios web o documentos relacionados.

Categorizar el contenido:

- Los usuarios deben poder asignar etiquetas o categorías a sus contenidos para facilitar la organización y búsqueda.
- Se debe permitir la creación y gestión de categorías personalizadas.

Revisar y observar:

- Los usuarios con el mismo rol del usuario creador deben poder revisar y observar el contenido para verificar su precisión y calidad.

Comentar:

- Los usuarios deben poder agregar comentarios a los contenidos para intercambiar ideas y comentarios con otros usuarios.

Pre visualizar:

- Los usuarios deben tener la capacidad de pre visualizar cómo se verá el contenido antes de publicarlo.

Publicar:

- Debe existir una función que permita a los usuarios publicar su contenido para que esté disponible para otros usuarios o estudiantes.

Acceso Móvil:

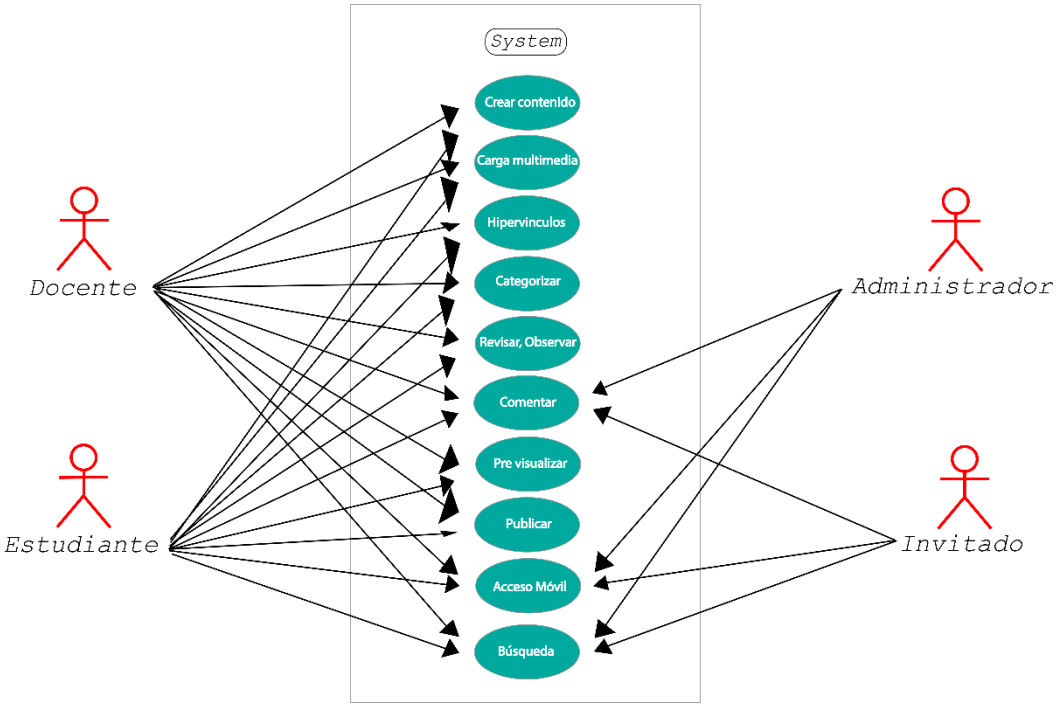
- Los usuarios deben poder acceder, crear y visualizar contenido a través de dispositivos móviles, como smartphones y tabletas.
- La interfaz debe ser responsive y adaptarse a diferentes tamaños de pantalla.

Búsqueda y Filtros:

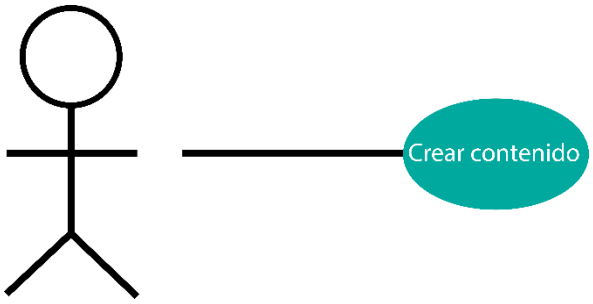
- Deben proporcionarse filtros para refinar las búsquedas por categorías, etiquetas, fecha de publicación y otros criterios relevantes.

Casos de Uso

Diagrama de caso de uso



Diagramas de Flujo de Casos de Uso



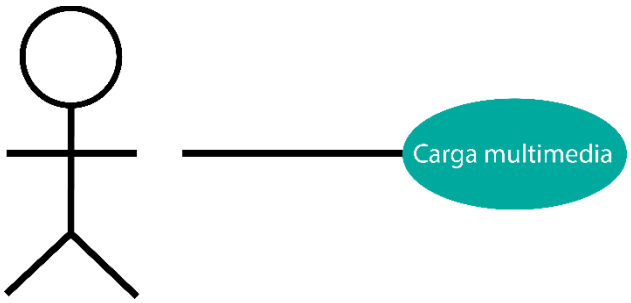
Prioridad: Requerido Versión: 1 Complejidad: Media CC: Crear Contenido	1. Crear Contenido Flujo: Crear Contenido Prueba: Seleccionar crear cocntenido.

Descripción detallada de cada caso de uso

CASO No. 1 Crear contenido

ID:	CU-1	
Nombre	Crear Contenido	
Actores	Docente	
Objetivo	Este caso debe permitir crear un contenido	
Urgencia	5	
Esfuerzo	4	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario ha accedido al módulo de creación de contenidos. ➢ Se han proporcionado las herramientas de creación necesarias, como el editor de contenido y las opciones de personalización.	
Flujo Normal	Docente	Sistema
	Selecciona la opción "Crear Contenido" desde la interfaz del módulo de creación de contenidos.	
		Presenta al usuario un editor de contenido interactivo que permite la creación y edición.
	Utiliza el editor para agregar texto	
	Puede personalizar el contenido, como la organización de secciones, el formato y la apariencia visual.	
	Guarda el contenido en progreso en cualquier momento durante la creación.	
		Retorna un mensaje confirmando el guardado del contenido.
Flujo alternativo 1	Si el usuario necesita hacer cambios en el contenido después de la pre visualización, puede volver al editor y realizar modificaciones.	
Flujo alternativo 2		

Post-condiciones	El usuario ha creado contenido educativo dentro del módulo de creación de contenidos.	
	El contenido se encuentra en estado de borrador y puede ser editado y revisado antes de su publicación.	
Exepciones	Error de Conexión: Si se produce un error de conexión durante la creación de contenido, el sistema muestra un mensaje de error al usuario, como "Error de conexión. Verifique su conexión a Internet". El usuario tiene la opción de "Reintentar Conexión" o ponerse en contacto con el soporte técnico si el problema persiste.	

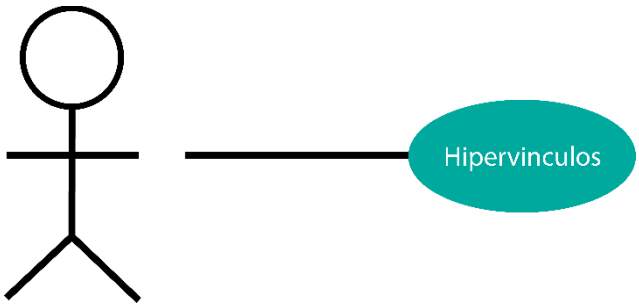


Prioridad: Opcional Versión: 1 Complejidad: Media CC: Crear Contenido IM: Insertar multimedia	2. Insertar multimedia Flujo: Insertar multimedia Prueba: Seleccionar crear cocntenido, Insertar multimedia. CC, IM

CASO No. 2 Carga multimedia.

ID:	CU-2	
Nombre	Carga multimedia	
Actores	Docente	
Objetivo	Permitir al usuario cargar y adjuntar archivos multimedia, como imágenes, videos y audio, al contenido que está creando dentro del módulo de creación de contenidos.	
Urgencia	3	
Esfuerzo	2	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario ha accedido al módulo de creación de contenidos. ➢ Se ha iniciado el proceso de creación de contenido.	
Flujo Normal	Docente	Sistema
	Selecciona la opción para cargar archivos multimedia dentro del editor de contenido.	
		Presenta una interfaz para seleccionar y cargar archivos desde el dispositivo del usuario.

	Selecciona los archivos multimedia que desea cargar (por ejemplo, imágenes, videos o audio).	
		Procesa y almacena los archivos multimedia en la plataforma.
	Los archivos multimedia cargados se integran en el contenido en el lugar deseado dentro del editor.	
Flujo alternativo 1	El usuario puede cancelar la carga de archivos en cualquier momento antes de confirmarla.	
	Si el usuario intenta cargar un tipo de archivo no admitido, el sistema muestra un mensaje de error y evita la carga del archivo no válido.	
Flujo alternativo 2		
Post-condiciones	El usuario ha cargado archivos multimedia en el contenido que está creando.	
	Los archivos multimedia se integran en el contenido y están disponibles para su visualización o reproducción.	
Exepciones	Formato de Archivo no Válido: Si el usuario intenta cargar un tipo de archivo no admitido, el sistema muestra un mensaje de error y evita la carga del archivo no válido.	

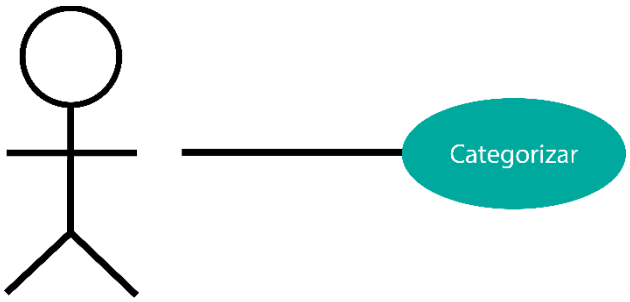


Prioridad: Opcional Versión: 1 Complejidad: Media CC: Crear Contenido IH: Insertar Hipervinculos	3. Insertar Hipervinculos Flujo: Insertar Hipervinculos Prueba: Seleccionar crear cocntenido, Insertar Hipervinculos. CC, IH

CASO No. 3 Inserción de Hipervínculos

ID:	CU-3	
Nombre	Inserción de Hipervínculos	
Actores	Docente / Estudiante	
Objetivo	Permitir al usuario insertar hipervínculos en el contenido que está creando, lo que le permite enlazar a recursos externos, como sitios web, documentos relacionados u otros contenidos dentro de la plataforma.	
Urgencia	2	
Esfuerzo	2	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario ha accedido al módulo de creación de contenidos. ➢ Se ha iniciado el proceso de creación de contenido.	
Flujo Normal	Docente	Sistema
	Selecciona la opción de inserción de hipervínculos en el editor de contenido.	
		Presenta una interfaz que permite al usuario ingresar la URL del recurso externo o seleccionar un recurso dentro de la plataforma.

	Especifica el texto de anclaje o descripción del hipervínculo.	
		Crea el hipervínculo y lo integra en el contenido en el lugar deseado.
Flujo alternativo 1	Si el usuario ingresa una URL no válida o incorrecta, el sistema muestra un mensaje de error y evita la creación del hipervínculo.	
Flujo alternativo 2		
Post-condiciones	El usuario ha insertado hipervínculos en el contenido que está creando.	
	Los hipervínculos funcionan correctamente y enlazan a los recursos externos o contenidos relacionados.	
Exepciones	URL no Válida: Si el usuario ingresa una URL no válida o incorrecta, el sistema muestra un mensaje de error y evita la creación del hipervínculo.	

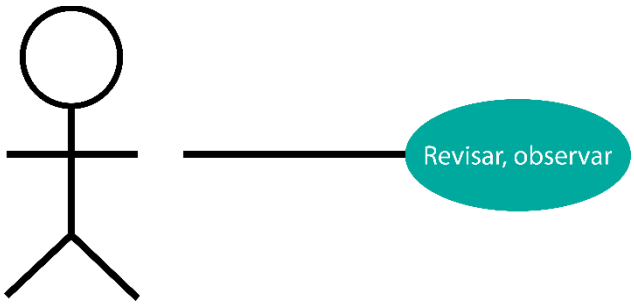


Prioridad: Opcional Versión: 1 Complejidad: Baja CC: Crear Contenido CT: Categotizar	4. Categotizar Flujo: Categotizar Prueba: Seleccionar crear cocntenido, categorizar, agregar nombre de la categoria. CC, CT

CASO No. 4 Categorizar Contenido

ID:	CU-4	
Nombre	Categorizar Contenido	
Actores	Docente	
Objetivo	Permitir al usuario asignar etiquetas o categorías a los contenidos que está creando, lo que facilita la organización y la búsqueda de los materiales de aprendizaje.	
Urgencia	2	
Esfuerzo	1	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario ha accedido al módulo de creación de contenidos. ➢ Se ha iniciado el proceso de creación de contenido.	
Flujo Normal	Docente	Sistema
	Selecciona la opción de categorización en el editor de contenido.	
		Muestra una interfaz que permite al usuario seleccionar o crear etiquetas o categorías para el contenido.
	Asigna una o varias etiquetas a su contenido.	
		Almacena la información de categorización en la plataforma y asocia el contenido a las etiquetas seleccionadas.
Flujo alternativo 1	Si el usuario intenta asignar una etiqueta que ya existe en la plataforma, el sistema la asocia al contenido existente en esa categoría.	
Flujo alternativo 2		
Post-condiciones	El usuario ha categorizado el contenido que está creando, facilitando su organización y recuperación.	

Exepciones	Etiqueta ya Existe: Si el usuario intenta asignar una etiqueta que ya existe en la plataforma, el sistema la asocia al contenido existente en esa categoría.	

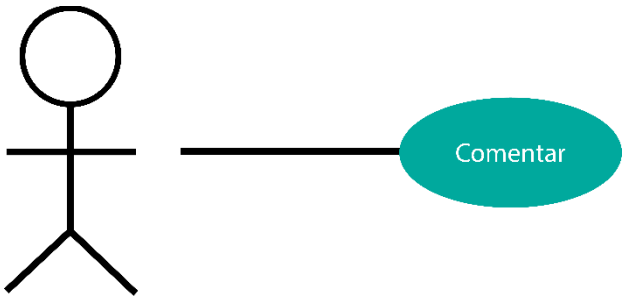


Prioridad: Opcional Versión: 1 Complejidad: Media BF: Busqueda, Filtro RV: Revisar	5. Revisar Flujo: Revisar Prueba: Seleccionar cocntenido creado, revisar contenido, agregar la observación. BC, RV

CASO No. 5 Revisiones y Observaciones

ID:	CU-5	
Nombre	Crear Contenido	
Actores	Docente	
Objetivo	Permitir a los docentes realizar revisiones y observaciones en el contenido creado por otros docentes o colaboradores. Esto es especialmente relevante en contextos donde solo los docentes pueden realizar correcciones.	
Urgencia	2	
Esfuerzo	1	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario tiene el rol de docente. ➢ Existe contenido creado por otros docentes o colaboradores.	
Flujo Normal	Docente	Sistema
	Accede al contenido que desea revisar.	
		Presenta una interfaz que permite al docente realizar observaciones y correcciones en el contenido.
	Agrega observaciones específicas y realiza las correcciones necesarias en el contenido.	
		Registra las observaciones y correcciones realizadas por el docente.
Flujo alternativo 1		
Flujo alternativo 2		
Post-condiciones	El contenido ha sido revisado y corregido por el docente.	
	Las observaciones y correcciones se registran y pueden ser revisadas por otros usuarios.	
Exepciones	Error de Conexión: Si se produce un error de conexión al realizar observaciones o correcciones, el sistema muestra un mensaje de	

	error. El docente puede "Reintentar Conexión" o ponerse en contacto con el soporte técnico si el problema persiste.	

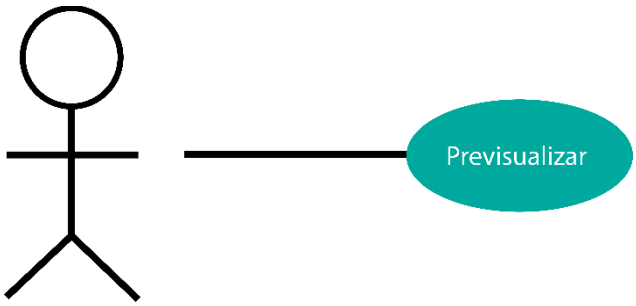


Prioridad: Opcional Versión: 1 Complejidad: Baja BF: Busqueda, Filtro CM: Comentar	6. Comentar Flujo: Comentar Prueba: Seleccionar cocntenido creado, comentar. BF, CM

CASO No. 6 Comentar en los Contenidos

ID:	CU-6	
Nombre	Comentar en los Contenidos	
Actores	Cualquier Usuario o Invitado	
Objetivo	Permitir a cualquier usuario, ya sea registrado en la plataforma o un invitado, agregar comentarios y participar en discusiones relacionadas con el contenido.	
Urgencia	2	
Esfuerzo	1	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma (en el caso de usuarios registrados). ➢ El contenido en el que se desea comentar está disponible para su visualización.	
Flujo Normal	Docente	Sistema
	Accede al contenido en el que desea dejar un comentario.	
		Presenta una interfaz que permite al usuario agregar un comentario en una sección de comentarios asociada al contenido.
	Escribe su comentario y lo publica en la sección de comentarios.	
	Otros usuarios pueden ver y responder a los comentarios publicados.	
Flujo alternativo 1	Los comentarios pueden ser moderados por un administrador o por las políticas de la plataforma para evitar contenido inapropiado.	
Flujo alternativo 2	Los comentarios pueden ser moderados por un administrador o por las políticas de la plataforma para evitar contenido inapropiado.	

Post-condiciones	El usuario ha publicado un comentario en el contenido.	
	Se ha creado una discusión relacionada con el contenido en la sección de comentarios.	
Exepciones	Contenido Moderado: Los comentarios pueden ser moderados por un administrador o por las políticas de la plataforma para evitar contenido inapropiado.	

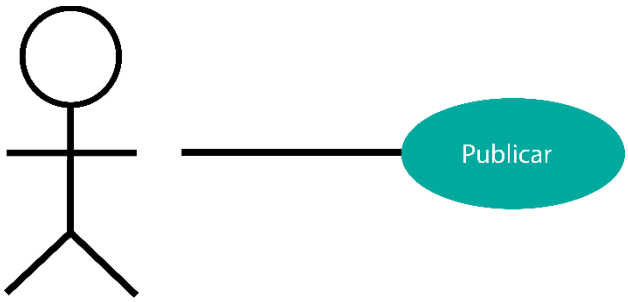


Prioridad: Opcional Versión: 1 Complejidad: Baja CC: Crear contenido PV: Pre visualizar	7. Pre visualizar Flujo: Pre visualizar Prueba: Crear contenido, previsualizar. CC, PV

CASO No. 7 Pre visualizar los Contenidos

ID:	CU-7	
Nombre	Pre visualizar los Contenidos	
Actores	Docente	
Objetivo	Permitir al usuario pre visualizar el contenido que está creando antes de publicarlo, lo que le brinda la oportunidad de verificar su apariencia y contenido antes de compartirlo con otros usuarios.	
Urgencia	3	
Esfuerzo	2	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario ha accedido al módulo de creación de contenidos. ➢ Se ha iniciado el proceso de creación de contenido.	
Flujo Normal	Docente	Sistema
	Selecciona la opción de pre visualización en el editor de contenido.	
		Muestra una representación del contenido tal como se verá una vez publicado.
	Puede revisar el diseño, la estructura y el contenido del material educativo.	
	Puede realizar cambios y ajustes en el contenido si es necesario.	
	Tiene la opción de regresar al modo de edición si desea realizar modificaciones adicionales.	
Flujo alternativo 1	Si el usuario no realiza cambios o ajustes en la pre visualización, puede proceder a publicar el contenido.	
Flujo alternativo 2		

Post-condiciones	El usuario ha pre visualizado el contenido y tiene la opción de realizar modificaciones antes de la publicación.	
Excepciones	Error de Pre visualización: Si se produce un error durante la pre visualización, el sistema muestra un mensaje de error. El usuario puede "Reintentar Pre visualización" o ponerse en contacto con el soporte técnico si el problema persiste.	

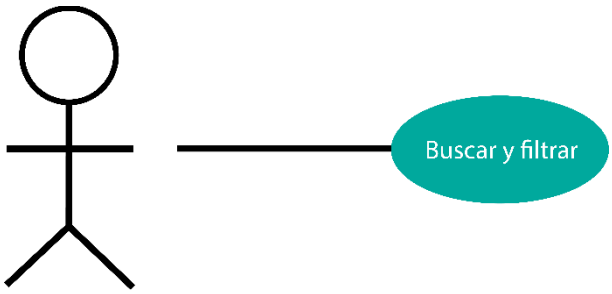


Prioridad: Requerido Versión: 1 Complejidad: Media CC: Crear contenido PB: Publicar	8. Publicar Flujo: Publicar Prueba: Crear contenido, publicar. CC, PB

CASO No. 8 Publicar Contenido

ID:	CU-8	
Nombre	Publicar Contenido	
Actores	Docente	
Objetivo	Permitir al usuario publicar el contenido educativo que ha creado en la plataforma, lo que lo hace accesible para otros usuarios, como alumnos, docentes y colaboradores.	
Urgencia	5	
Esfuerzo	4	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma. ➢ El usuario ha accedido al módulo de creación de contenidos. ➢ El contenido ha sido creado y pre visualizado (si se requiere).	
Flujo Normal	Docente	Sistema
	Selecciona la opción para publicar el contenido que ha creado.	
		Muestra una interfaz que permite al usuario especificar la visibilidad del contenido (público, privado, solo para ciertos usuarios, etc.).
	Confirma la publicación del contenido.	
	El contenido se hace accesible para otros usuarios según las preferencias de visibilidad especificadas.	
Flujo alternativo 1	Si el usuario no ha pre visualizado el contenido, se le puede ofrecer la opción de hacerlo antes de la publicación.	
Flujo alternativo 2		
Post-condiciones	El contenido ha sido publicado y se encuentra disponible para otros usuarios según las preferencias de visibilidad establecidas.	

Exepciones	Error de Publicación: Si se produce un error durante la publicación, el sistema muestra un mensaje de error. El usuario puede "Reintentar Publicación" o ponerse en contacto con el soporte técnico si el problema persiste.	



Prioridad: Opcional Versión: 1 Complejidad: Baja BF: Busqueda, Filtro	10. Busqueda, Filtro Flujo: Busqueda, Filtro Prueba: Buscar contenido BF

CASO No. 10 Buscar y Filtrar Contenidos

ID:	CU-10	
Nombre	Buscar y Filtrar Contenidos	
Actores	Cualquier Usuario o Invitado	
Objetivo	Permitir a los usuarios buscar y filtrar contenidos en la plataforma, lo que facilita la localización de materiales de aprendizaje específicos o relevantes.	
Urgencia	1	
Esfuerzo	1	
Pre-condiciones	➢ El usuario ha iniciado sesión en la plataforma (en el caso de usuarios registrados). ➢ El usuario tiene acceso a la función de búsqueda y filtrado de contenidos.	
Flujo Normal	Docente	Sistema
	Selecciona la opción de búsqueda y filtrado en la interfaz de la plataforma.	
		Presenta una interfaz que permite al usuario ingresar términos de búsqueda y aplicar filtros según diferentes criterios (por ejemplo,

		tema, tipo de contenido, fecha de publicación, etc.).
	Ingresa los términos de búsqueda y selecciona los filtros deseados.	
		Realiza la búsqueda y muestra los resultados que coinciden con los términos de búsqueda y filtros aplicados.
	Puede seleccionar un resultado para acceder al contenido o refinar aún más la búsqueda y los filtros si es necesario.	
Flujo alternativo 1	Si la búsqueda no arroja resultados, el sistema debe proporcionar un mensaje informativo y permitir al usuario ajustar los términos de búsqueda.	
Flujo alternativo 2		
Post-condiciones	El usuario ha realizado una búsqueda y ha encontrado los contenidos deseados.	
Exepciones	Búsqueda sin Resultados: Si la búsqueda no arroja resultados, el sistema debe proporcionar un mensaje informativo y permitir al usuario ajustar los términos de búsqueda.	

Prioridad de Requerimientos

A partir del análisis de requerimientos, funcionalidades y el proceso de design thinking, se concreta la siguiente matrix 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)

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

https://asana.com/es/resources/priority-matrix

4. Requisitos No Funcionales

1. **Rendimiento:**

   - El sistema debe ser capaz de manejar múltiples usuarios creando contenido simultáneamente sin degradación del rendimiento.

   - Los tiempos de carga de la plataforma y las vistas previas de contenidos deben ser rápidos, incluso en conexiones de red más lentas.

2. **Disponibilidad:**

   - El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad programado mínimo para el mantenimiento.

   - Se debe implementar un sistema de copias de seguridad y recuperación de datos para garantizar la disponibilidad de contenidos en caso de fallos.

3. **Seguridad:**

   - Se debe implementar una autenticación segura para los usuarios.

   - Los datos de los usuarios y los contenidos deben estar protegidos mediante cifrado y medidas de seguridad adecuadas.

   - Se debe establecer un sistema de control de acceso para garantizar que solo los usuarios autorizados puedan editar y publicar contenidos.

4. **Escalabilidad:**

   - El sistema debe ser escalable para adaptarse al crecimiento de usuarios y contenidos sin problemas.

   - Debe ser posible agregar recursos de hardware y software según sea necesario.

5. **Usabilidad:**

   - La interfaz de usuario debe ser intuitiva y fácil de usar para que los docentes y usuarios puedan crear contenidos sin necesidad de capacitación adicional.

   - Debe ser accesible para personas con discapacidades, cumpliendo con las pautas de accesibilidad web.

6. Compatibilidad:

- El módulo de creación de contenidos debe ser compatible con una amplia variedad de navegadores web y sistemas operativos.
- Debe funcionar en dispositivos móviles y de escritorio.

7. Mantenibilidad:

- El sistema debe ser fácil de mantener y actualizar sin causar interrupciones en el servicio.
- El código debe estar bien documentado para facilitar futuras modificaciones y mejoras.

8. Desempeño y Eficiencia:

- El sistema debe utilizar eficientemente los recursos del servidor y minimizar el consumo de ancho de banda.
- Las funciones deben ejecutarse de manera rápida y eficiente.

9. Cumplimiento Normativo:

- El sistema debe cumplir con las regulaciones y normativas aplicables, como la protección de datos y los derechos de autor.

10. Internacionalización:

- El sistema debe admitir múltiples idiomas y regiones, permitiendo la creación y visualización de contenidos en diferentes lenguajes.

Requisitos de Desempeño

1. **Tiempo de Carga:**

    - El sistema debe cargar la interfaz de creación de contenidos en un máximo de 2 segundos para proporcionar una experiencia de usuario fluida.

2. **Tiempo de Guardado:**

    - El sistema debe permitir a los usuarios guardar los cambios en el contenido de forma instantánea o en menos de 1 segundo.

3. **Capacidad de Usuarios Concurrentes:**

    - El sistema debe ser capaz de admitir al menos 100 docentes o usuarios creando contenidos simultáneamente sin afectar el rendimiento.

4. **Capacidad de Almacenamiento:**

    - El sistema debe ser capaz de manejar al menos 1 terabyte de contenidos creados por los usuarios.

5. **Eficiencia de Búsqueda y Filtrado:**

    - Las búsquedas y filtros de contenidos deben entregar resultados en menos de 2 segundos.

6. **Tiempos de Respuesta en Dispositivos Móviles:**

    - Las vistas previas y la creación de contenidos deben funcionar de manera eficiente en dispositivos móviles, con tiempos de respuesta de 3 segundos o menos.

7. **Escalabilidad:**

    - El sistema debe ser escalable y capaz de manejar un aumento del 50% en usuarios y contenidos en un período de 1 año.

8. **Optimización de Imágenes y Multimedia:**

    - El sistema debe optimizar automáticamente las imágenes y archivos multimedia cargados para garantizar una carga rápida de contenidos.

9. **Uso Eficiente de Recursos del Servidor:**

- El sistema debe utilizar eficientemente los recursos del servidor, minimizando el consumo de CPU y memoria.

10. Disponibilidad del 99.9%:

- El sistema debe estar disponible el 99.9% del tiempo, lo que permite un tiempo de inactividad programado de hasta 8.76 horas al año.

Requisitos de Seguridad

1. **Autenticación Segura:**
   - Los usuarios deben autenticarse de manera segura utilizando contraseñas fuertes. Se debe implementar un sistema de bloqueo de cuentas después de un número determinado de intentos fallidos.
2. **Control de Acceso:**
   - El sistema debe asignar roles y permisos específicos a los usuarios para controlar el acceso a las funciones y los contenidos. Solo los usuarios autorizados pueden editar y publicar contenidos.
3. **Cifrado de Datos:**
   - Todos los datos en tránsito y en reposo deben estar cifrados. Esto incluye la comunicación entre el navegador del usuario y el servidor, así como el almacenamiento de contraseñas y datos sensibles.
4. **Protección contra Amenazas Comunes:**
   - El sistema debe estar protegido contra amenazas comunes, como ataques de inyección de SQL, ataques de fuerza bruta y ataques de seguridad web.
5. **Auditoría y Registro de Eventos:**
   - Se debe mantener un registro de eventos y actividades en el sistema, lo que permite realizar un seguimiento de las acciones de los usuarios y detectar comportamientos sospechosos.
6. **Seguridad de Contenidos:**
   - Los contenidos creados por los usuarios deben ser escaneados en busca de malware y contenido malicioso antes de ser publicados.
7. **Política de Contraseñas:**
   - Se debe implementar una política de contraseñas que requiera contraseñas seguras y su cambio periódico.
8. **Recuperación de Cuenta:**
   - Debe existir un proceso de recuperación de cuentas seguro para que los usuarios puedan restablecer sus contraseñas o recuperar sus cuentas de manera segura.
9. **Cumplimiento Normativo:**
   - El sistema debe cumplir con regulaciones y normativas aplicables, como las relacionadas con la protección de datos y la privacidad.
10. **Protección de Datos de Usuario:**

- Los datos personales de los usuarios, como nombres, correos electrónicos y otra información confidencial, deben estar protegidos y no compartidos con terceros sin consentimiento.

11. Pruebas de Seguridad:

- Se deben realizar pruebas de seguridad regulares para identificar y abordar posibles vulnerabilidades en el sistema.

12. Seguridad en las Comunicaciones:

- Las comunicaciones entre el navegador del usuario y el servidor deben estar protegidas mediante HTTPS para prevenir la interceptación de datos.

Requisitos de Usabilidad

11. **Interfaz Intuitiva:**
    - La interfaz de usuario debe ser intuitiva y fácil de usar, de modo que los usuarios puedan aprender a utilizarla rápidamente.
12. **Navegación Sencilla:**
    - La navegación dentro del módulo debe ser sencilla y lógica, con una estructura de menús clara.
13. **Retroalimentación al Usuario:**
    - El sistema debe proporcionar retroalimentación inmediata al usuario en respuesta a sus acciones, como la confirmación de que un contenido se ha guardado correctamente.
14. **Personalización:**
    - Los usuarios deben poder personalizar su experiencia, como ajustar la configuración de la interfaz y definir preferencias.
15. **Accesibilidad:**
    - La plataforma debe ser accesible para personas con discapacidades, cumpliendo con las pautas de accesibilidad web y brindando soporte para lectores de pantalla.
16. **Documentación y Ayuda:**
    - Debe proporcionarse documentación y recursos de ayuda, como tutoriales, guías o FAQ, para asistir a los usuarios en el uso del módulo.
17. **Compatibilidad:**
    - El módulo debe ser compatible con una variedad de navegadores web y sistemas operativos.
18. **Pruebas de Usabilidad:**
    - Deben realizarse pruebas de usabilidad con usuarios reales para identificar y abordar posibles problemas de usabilidad.
19. **Feedback de Usuarios:**
    - Se debe proporcionar a los usuarios la posibilidad de proporcionar comentarios y sugerencias para mejorar la usabilidad.
20. **Diseño Responsivo:**
    - La interfaz de usuario debe ser responsiva y adaptable a diferentes tamaños de pantalla, incluyendo dispositivos móviles y tabletas.
21. **Flujo de Trabajo Eficiente:**
    - El módulo debe permitir a los usuarios crear contenidos de manera eficiente y con un flujo de trabajo optimizado.
22. **Consistencia en el Diseño:**
    - El diseño de la interfaz debe ser consistente en todas las secciones del módulo.
23. **Etiquetado Claro:**

- Los elementos de la interfaz, como botones y menús, deben tener etiquetas claras y comprensibles.

24. Facilidad de Aprendizaje:

- El módulo debe ser fácil de aprender, incluso para usuarios nuevos.

25. Tiempo de Respuesta Rápido:

- El tiempo de respuesta del sistema, como la carga de vistas previas y la creación de contenidos, debe ser rápido para mantener a los usuarios comprometidos.

Requisitos de Escalabilidad

1. **Capacidad de Usuarios Concurrentes:**

   - El sistema debe ser escalable y capaz de admitir un aumento sustancial de usuarios concurrentes sin afectar el rendimiento.

2. **Capacidad de Almacenamiento:**

   - El sistema debe poder escalar para manejar un crecimiento significativo en la cantidad de contenidos creados y almacenados.

3. **Balanceo de Carga:**

   - Debe implementarse un mecanismo de balanceo de carga que distribuya el tráfico de usuarios de manera equitativa entre múltiples servidores para evitar la sobrecarga.

4. **Escalabilidad Horizontal:**

   - El sistema debe permitir la adición de servidores y recursos de hardware a medida que sea necesario para satisfacer la demanda.

5. **Escalabilidad Vertical:**

   - Debe ser posible aumentar la capacidad de servidores individuales para manejar una mayor carga de trabajo.

6. **Elasticidad:**

   - El sistema debe ser elástico, lo que significa que debe escalar automáticamente hacia arriba o hacia abajo según las necesidades, sin intervención manual.

7. **Almacenamiento Distribuido:**

   - Los datos y contenidos deben poder almacenarse en sistemas de almacenamiento distribuido para facilitar la escalabilidad.

8. **Optimización de Bases de Datos:**

- Deben implementarse prácticas de optimización de bases de datos para garantizar que puedan manejar grandes volúmenes de datos sin problemas.

9. Pruebas de Escalabilidad:

- Deben realizarse pruebas de escalabilidad para evaluar el rendimiento del sistema bajo cargas pesadas y verificar su capacidad de crecimiento.

10. Plan de Escalabilidad:

- Debe desarrollarse un plan detallado que establezca cómo el sistema se escalara a medida que aumente la demanda, incluyendo el momento de agregar recursos adicionales.

11. Monitorización de Recursos:

- El sistema debe ser capaz de monitorear el uso de recursos, como la CPU y la memoria, para tomar decisiones informadas sobre la escalabilidad.

12. Escalabilidad Geográfica:

- Si se planea una expansión geográfica, el sistema debe poder replicar y escalarse en diferentes ubicaciones geográficas.

5. Modelado E/R

Diagrama de Entidad-Relación

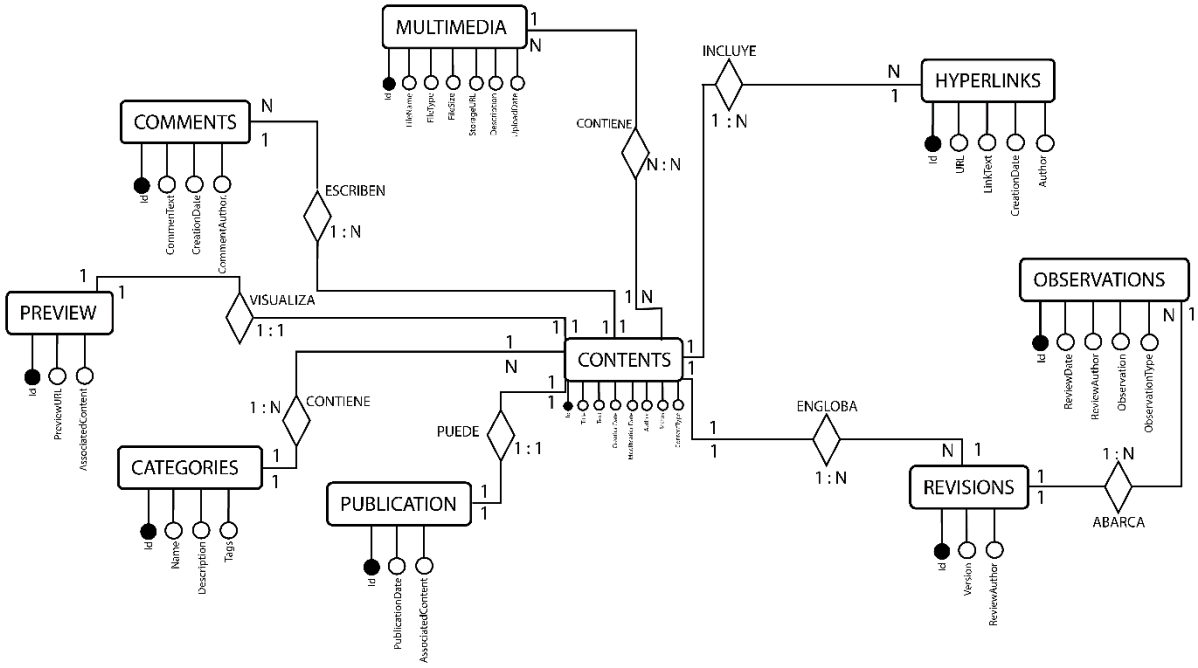
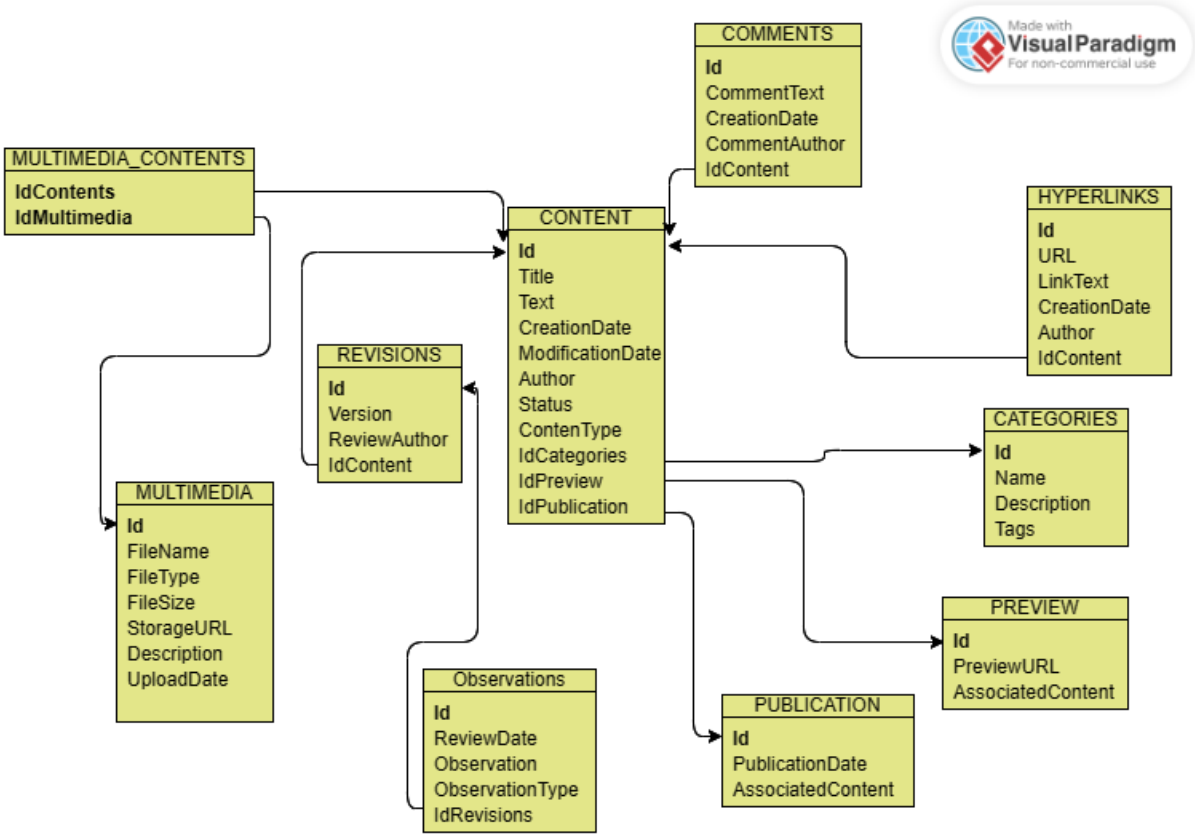


Diagrama Relacional



Script de modelo relacional

```
Table content {
  _id varchar [primary key]
  title varchar
  text varchar
  author varchar
  creationDate timestamp
  modificationDate timestamp
  status varchar

  category_id varchar [ref: > categories._id]
  preview_id varchar [ref: > preview_views._id]
  publication_id varchar [ref: > publications._id]
}

Table multimedia {
  _id varchar [primary key]
  file_name varchar
  file_type varchar
  file_size integer
  storage_url varchar
  description varchar
  upload_date timestamp
  content_id varchar [ref: > content._id]
}

Table hyperlinks {
  _id varchar [primary key]
  url varchar
  link_text varchar
  creation_date timestamp
  author varchar
  content_id varchar [ref: > content._id]
}

Table comments {
  _id varchar [primary key]
  comment_text varchar
  creation_date timestamp
  author varchar
  content_id varchar [ref: > content._id]
}

Table categories {
  _id varchar [primary key]
  name varchar
  description varchar
  tags varchar
  vvhvh va
}
```

```
Table revisions {
  _id varchar [primary key]
  version varchar
  review_author varchar
  content_id varchar [ref: > content._id]
}

Table observations {
  _id varchar [primary key]
  review_date timestamp
  review_author varchar
  observation varchar
  observation_type varchar
  revision_id varchar [ref: > revisions._id]
}

Table preview_views {
  _id varchar [primary key]
  preview_url varchar
  associated_content varchar
}

Table publications {
  _id varchar [primary key]
  publish_start_date timestamp
  associated_content varchar
}
```

Descripción de Entidades y Relaciones

Entidades:

1. Entidad: Contenidos

- ***Atributos***: Id, Título, Texto, Fecha de creación, Fecha de modificación, Autor, Estado (publicado, borrador, etc.), Tipo de contenido.

2. Entidad: Multimedia

- ***Atributos***: Id, Nombre del archivo, Tipo de archivo, Tamaño del archivo, URL de almacenamiento, Descripción, Fecha de carga.

3. Entidad: Hipervínculos

- ***Atributos***: Id, URL, Texto del enlace, Fecha de creación, Autor.

4. Entidad: Revisiones

- ***Atributos***: Id, Versión, Autor de revisión.

5. Entidad: Observaciones

- ***Atributos***: Id, Fecha de revisión, Autor de la revisión, Observación, Tipo de observación.

6. Entidad: Categorías o Secciones

- ***Atributos***: Id, Nombre, Descripción, Etiquetas.

7. Entidad: Comentarios

- ***Atributos***: Id, Texto del comentario, Fecha de creación, Autor del comentario.

8. Entidad: Publicación

- ***Atributos***: Id, Fecha y hora de inicio de publicación, Contenido asociado.

9. Entidad: Vistas Previa

- ***Atributos***: Id, URL de vista previa, Contenido asociado.

Relaciones:

Entidad: Contenidos

Relaciones:

Uno a muchos con "Multimedia."

Uno a muchos con "Hipervínculos."

Uno a muchos con "Revisiones."

Uno a muchos con "Observaciones."

Muchos a uno con "Categorías o Secciones."

Uno a muchos con "Comentarios."

Uno a uno con "Publicación."

Uno a uno con "Vistas Previa."

Entidad: Multimedia

Relaciones:

Muchos a uno con "Contenidos."

Entidad: Hipervínculos

Relaciones:

Muchos a uno con "Contenidos."

Entidad: Revisiones

Relaciones:

Muchos a uno con "Contenidos."

Entidad: Observaciones

Relaciones:

Muchos a uno con "Contenidos."

Entidad: Categorías o Secciones

Relaciones:

Uno a muchos con "Contenidos."

Entidad: Comentarios

Relaciones:

Muchos a uno con "Contenidos."

Entidad: Publicación

Relaciones:

Uno a uno con "Contenidos."

Entidad: Vistas Previa

Relaciones:

Uno a uno con "Contenidos."

.

Reglas de Integridad Referencial

1. **Entidad: Contenidos:**
   - Clave Primaria: Puede ser un campo como "IDContenido" que es único para cada contenido.
   - Clave Externa: Si los contenidos se relacionan con otras entidades (por ejemplo, categorías), puede haber claves externas que se relacionen con las claves primarias de esas entidades.
2. **Entidad: Multimedia:**
   - Clave Primaria: Por ejemplo, "IDMultimedia."
   - Clave Externa: Puede estar relacionada con la entidad "Contenidos" para asociar multimedia específica con contenidos.
3. **Entidad: Hipervínculos:**
   - Clave Primaria: Un campo como "IDHipervinculo."
   - Clave Externa: Puede relacionarse con la entidad "Contenidos" para vincular hipervínculos a contenidos específicos.
4. **Entidad: Revisiones:**
   - Clave Primaria: Por ejemplo, "IDRevision."
   - Clave Externa: Puede estar relacionada con la entidad "Contenidos" para vincular revisiones a contenidos específicos. Además, la regla de eliminación en cascada podría aplicarse si se desean eliminar revisiones relacionadas con un contenido eliminado.
5. **Entidad: Observaciones:**
   - Clave Primaria: Puede ser un campo como "IDObservacion."
   - Clave Externa: Generalmente, las observaciones se relacionarán con "Revisiones" o "Contenidos" y tendrán claves externas que apunten a las claves primarias de esas entidades.
6. **Entidad: Categorías o Secciones:**
   - Clave Primaria: Un campo como "IDCategoria" o "IDSeccion."
   - Clave Externa: Si los contenidos se asignan a categorías o secciones, se pueden usar claves externas en la entidad "Contenidos" para relacionarlos con las categorías o secciones correspondientes.
7. **Entidad: Comentarios:**
   - Clave Primaria: Puede ser un campo como "IDComentario."
   - Clave Externa: Los comentarios generalmente se relacionarán con "Contenidos" o "Usuarios" y tendrán claves externas que apunten a las claves primarias de esas entidades.
8. **Entidad: Publicación:**
   - Clave Primaria: Por ejemplo, "IDPublicacion."

- Clave Externa: Las publicaciones pueden estar relacionadas con "Contenidos" y tendrán claves externas que apunten a los contenidos correspondientes.

9. **Entidad: Vistas Previa:**
   - Clave Primaria: Puede ser un campo como "IDVistaPrevia."
   - Clave Externa: Las vistas previas generalmente se relacionarán con "Contenidos" y tendrán claves externas que apunten a los contenidos a los que pertenecen.

Colecciones (NoSQL)

Categories {

_id: string;

name: string;

description: string;

tags: object;}

Comments {

_id: string;

comment_text: string;

creation_date: Date;

author: string;}

Contents {

_id: string;

title: string;

text: string;

author: string;

creationdate: Date;

modificationdate: Date;

status: string;}

Hyperlinks {

_id: string;

url: string;

link_text: string;

creation_date: Date;

author: string;}

Multimedia {

_id: string;

file_name: string;

file_type: string;

file_size: number;

storage_url: string;

description: string;

upload_date: Date;}

Observations {

_id: string;

review_date: Date;

review_author: string;

observation: string;

observation_type: string;}

Preview {

_id: string;

preview_url: string;

associated_content: object;}

Publications {

_id: string;

publish_start_date: Date;

associated_content: object;}

Revisions {

_id: string;

version: string;

review_author: string;}

6. Anexos

Diagramas Adicionales

Referencias

Etapa 2: Persistencia de Datos con Backend

7. Introducción

La segunda etapa de persistencia de datos con backend desempeña un papel fundamental en el desarrollo de nuestro software educativo. En esta fase, nos adentramos en el corazón del sistema, donde se conciben y gestionan los contenidos que serán la columna vertebral de la experiencia de aprendizaje. En esta sección introductoria, exploraremos el propósito y el alcance de esta etapa crítica, delineando claramente los objetivos que buscamos alcanzar y los límites que definirán nuestra labor.

Propósito de la Etapa

El propósito fundamental de esta etapa de persistencia de datos es la creación, gestión y almacenamiento eficiente de los contenidos educativos que enriquecerán la experiencia de nuestros usuarios. Aquí se forjarán las bases para brindar un entorno de aprendizaje interactivo y personalizado. La etapa de persistencia de datos es el eslabón clave que garantiza la disponibilidad, accesibilidad y actualización de los recursos pedagógicos, lo que contribuirá significativamente al éxito de nuestro software educativo.

Alcance de la Etapa

En cuanto al alcance, esta etapa abarcará la definición de la estructura de datos, la implementación de un sistema de gestión de contenidos (CMS) adaptado a las necesidades educativas, la interacción con los usuarios para la creación y modificación de contenidos, y la integración con otros componentes del software, como la interfaz de usuario y las funcionalidades de seguimiento del progreso del estudiante. Asimismo, abordaremos la seguridad, la escalabilidad y la eficiencia en el almacenamiento de datos para garantizar un funcionamiento óptimo.

Con esta visión general, estamos listos para adentrarnos en los detalles de la segunda etapa de persistencia de datos con backend y trabajar en conjunto para construir una plataforma educativa de calidad que cumpla con nuestras metas y expectativas.

8. Diseño de la Arquitectura de Backend

Descripción de la Arquitectura Propuesta

La arquitectura del backend se basa en el marco de trabajo Nest.js, que sigue el patrón Modelo-Vista-Controlador (MVC) y proporciona una estructura organizada y escalable para el desarrollo.

- **Capa de Presentación**: En esta capa, Nest.js maneja las solicitudes HTTP y las rutas a los controladores correspondientes, permitiendo la interacción con el cliente.

- **Capa de Lógica de Negocio**: Los servicios de Nest.js contienen la lógica de la aplicación, como la gestión de usuarios, contenidos y categorías.

- **Capa de Datos**: Aquí se gestionan las operaciones de lectura y escritura de datos en la base de datos, con modelos y esquemas definidos.

Tecnologías Específicas de Nest.js:

- Decoradores: Utilizados para definir rutas y metadatos en controladores y servicios.

- Módulos: Organizan y encapsulan componentes relacionados para modularidad y reutilización del código.

- Inyección de Dependencias: Proporciona servicios y componentes a las clases que los necesitan.

- Middleware: Agrega funcionalidades adicionales, como autenticación y autorización, a las rutas.

Escalabilidad: Nest.js permite manejar un mayor número de solicitudes con balanceadores de carga y escalabilidad en múltiples instancias.

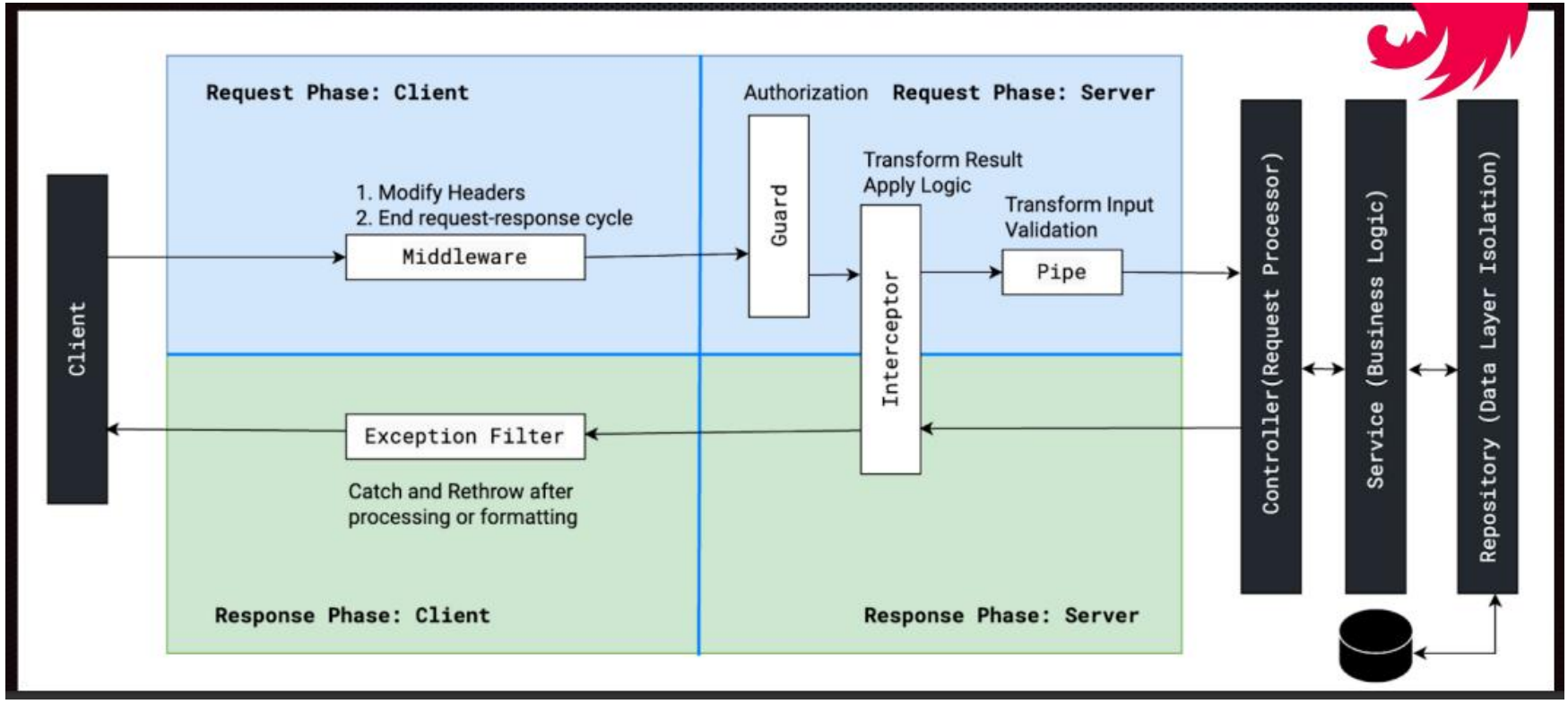
Caché: Implementación de una capa de caché para mejorar el rendimiento y reducir la carga en la base de datos.

Documentación de la API: Generación automática de documentación de API basada en controladores y rutas.

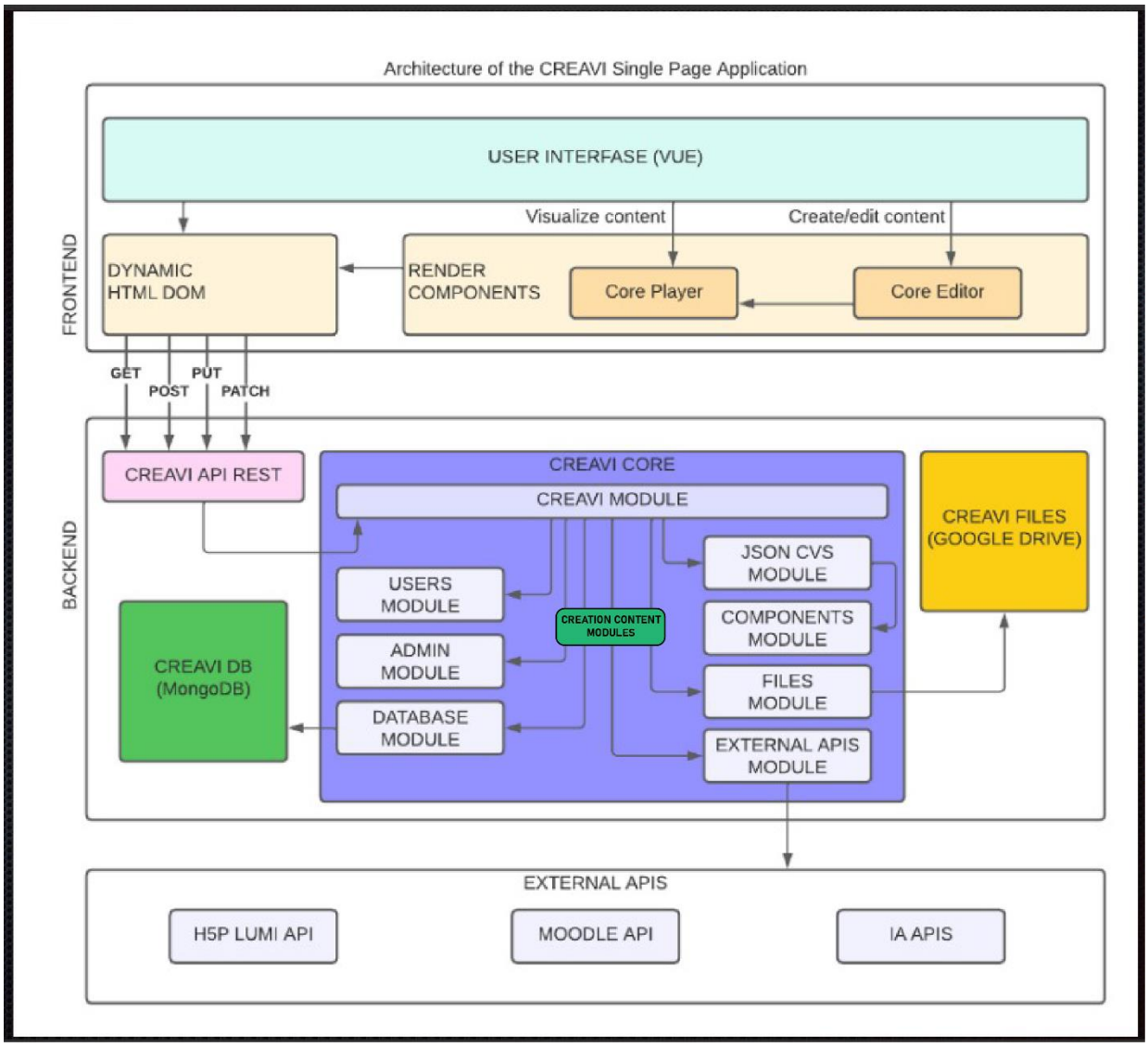
Monitorización y Registro: Uso de herramientas para supervisar el rendimiento y detectar problemas en tiempo real.

Seguridad: Aplicación de prácticas sólidas de seguridad, incluyendo protección contra ataques, autenticación segura y autorización adecuada.

Componentes del Backend



Diagramas de Arquitectura



9. Elección de la Base de Datos

Evaluación de Opciones (SQL o NoSQL)

En la evaluación de opciones para la base de datos de nuestro proyecto, hemos optado por utilizar una base de datos NoSQL. Esta elección se basa en una serie de consideraciones específicas que son fundamentales para el éxito de nuestra aplicación.

Justificación de la Elección

La elección de una base de datos NoSQL se justifica por varias razones cruciales. En primer lugar, una base de datos NoSQL nos permite una escalabilidad excepcional tanto vertical como horizontal. Dado que nuestra aplicación educativa tiene el potencial de experimentar un crecimiento significativo en términos de usuarios y datos, necesitamos una solución que nos permita aumentar la capacidad de almacenamiento y el rendimiento de manera eficiente, sin las limitaciones impuestas por los modelos de bases de datos relacionales convencionales.

Además, la flexibilidad inherente de las bases de datos NoSQL es esencial para nuestra aplicación. En el contexto educativo, los datos pueden variar en formato y estructura, y esta versatilidad es fundamental para gestionar recursos de aprendizaje que pueden incluir texto, imágenes, videos, cuestionarios y más

Diseño de Esquema de Base de Datos

Esquema de colección categories

TS categorie.schema.ts U

src > categories > schemas > TS categorie.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Categories extends Document {
    @Prop()
    _id: string;

    @Prop()
    name: string;

    @Prop()
    description: string;

    @Prop()
    tags: object;
}

export const CategoriesSchema = SchemaFactory.createForClass(Categories);
```

Esquema de colección comments

TS comments.schema.ts U

src > comments > schemas > TS comments.schema.ts > ...

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

@Schema({
    timestamps: true
})

export class comments extends Document {
    @Prop()
    _id: string;

    @Prop()
    comment_text: string;

    @Prop()
    creation_date: Date;

    @Prop()
    author: string;
}

export const CommentSchema = SchemaFactory.createForClass(comments);
```

Esuqema de colección contents

TS content.schema.ts U

src > contents > schemas > TS content.schema.ts > ...

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

@Schema({
    timestamps: true
})

export class contents extends Document {
    @Prop()
    _id: string;

    @Prop()
    title: string;

    @Prop()
    text: string;

    @Prop({type: mongoose.Schema.Types.ObjectId, ref: 'User'})
    author: string;

    @Prop()
    creationdate: Date;

    @Prop()
    modificationdate: Date;

    @Prop({type: String, enum:['DRAFT', 'PUBLISHED', 'DELETE'], default: 'DRAFT'})
    status: string;
}

export const ContentSchema = SchemaFactory.createForClass(contents);
```

Esuqema de colección hyperlinks

TS hyperlinks.schema.ts U

src > hyperlinks > schemas > TS hyperlinks.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Hyperlinks extends Document {
    @Prop()
    _id: string;

    @Prop()
    url: string;

    @Prop()
    link_text: string;

    @Prop()
    creation_date: Date;

    @Prop()
    author: string;
}

export const HyperlinksSchema = SchemaFactory.createForClass(Hyperlinks);
```

Esuqema de colección multimedia

TS multimedia.schema.ts U

src > multimedia > schemas > TS multimedia.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Multimedia extends Document {
    @Prop()
    _id: string;

    @Prop()
    file_name: string;

    @Prop()
    file_type: string;

    @Prop()
    file_size: number;

    @Prop()
    storage_url: string;

    @Prop()
    description: string;

    @Prop()
    upload_date: Date;
}

export const MultimediaSchema = SchemaFactory.createForClass(Multimedia);
```

Esuqema de colección observations

observations.schema.ts U

src > observations > schemas > observations.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Observations extends Document {
    @Prop()
    _id: string;

    @Prop()
    review_date: Date;

    @Prop()
    review_author: string;

    @Prop()
    observation: string;

    @Prop()
    observation_type: string;
}

export const ObservationsSchema = SchemaFactory.createForClass(Observations);
```

Esuqema de colección preview

preview.schema.ts U

src > preview > schemas > preview.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Preview extends Document {
    @Prop()
    _id: string;

    @Prop()
    preview_url: string;

    @Prop()
    associated_content: object;
}

export const PreviewSchema = SchemaFactory.createForClass(Preview);
```

Esuqema de colección publications

TS publication.schema.ts U

src > publication > schemas > TS publication.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Publicacion extends Document {
    @Prop()
    _id: string;

    @Prop()
    publish_start_date: Date;

    @Prop()
    associated_content: object;
}

export const PublicacionSchema = SchemaFactory.createForClass(Publicacion);
```

Esuqema de colección revisions

TS revision.schema.ts U

src > revisions > schemas > TS revision.schema.ts > ...

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

@Schema({
    timestamps: true
})
export class Revisions extends Document {
    @Prop()
    _id: string;

    @Prop()
    version: string;

    @Prop()
    review_author: string;
}

export const RevisionsSchema = SchemaFactory.createForClass(Revisions);
```

10.Implementación del Backend

Elección del Lenguaje de Programación

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft. Es un superconjunto tipado de JavaScript, lo que significa que añade características de tipado estático a JavaScript y compila a código JavaScript estándar que puede ser ejecutado en cualquier entorno que soporte JavaScript.

Características de TypeScript:

1. Tipado Estático:

TypeScript introduce el concepto de tipado estático, lo que significa que puedes declarar tipos de datos para variables, funciones, y otros elementos en tiempo de desarrollo. Esto ayuda a detectar errores de tipo antes de que el código se ejecute, facilitando la detección y corrección de errores durante el desarrollo.

2. Compilación a JavaScript:

TypeScript no se ejecuta directamente en los navegadores o entornos de ejecución de JavaScript. En su lugar, se compila a JavaScript. Esto significa que puedes aprovechar las características de TypeScript durante el desarrollo, pero el código resultante es compatible con cualquier navegador o entorno que admita JavaScript.

3. Orientado a Objetos:

TypeScript es un lenguaje orientado a objetos que permite la programación orientada a clases, interfaces y herencia. Estas características proporcionan una estructuración más organizada y modular del código.

4. Amplio Soporte para ECMAScript:

TypeScript sigue de cerca las especificaciones de ECMAScript y a menudo adelanta las características que aún no se han implementado en las versiones actuales de JavaScript. Esto permite utilizar las últimas características de JavaScript incluso antes de que estén disponibles en los navegadores.

5. Herramientas de Desarrollo Mejoradas:

Al tener tipado estático, TypeScript mejora la experiencia de desarrollo al proporcionar una mejor autocompletado, sugerencias de código y detección de errores en tiempo real en entornos de desarrollo integrados (IDE) como Visual Studio Code.

6. Compatibilidad con Bibliotecas y Marcos de Trabajo de JavaScript:

Puedes utilizar las bibliotecas y marcos de trabajo de JavaScript existentes en proyectos de TypeScript. TypeScript es compatible con el ecosistema de JavaScript, lo que facilita la integración con bibliotecas populares y marcos de trabajo.

7. Escalabilidad:

TypeScript es especialmente útil en proyectos grandes y complejos, ya que su sistema de tipos y otras características facilitan el mantenimiento, la refactorización y la escalabilidad del código.

Creación de la Lógica de Negocio

1. **Contenidos:** Esta colección puede almacenar entradas de contenido, como artículos, publicaciones o información principal de la plataforma. Contiene los detalles esenciales del contenido, como título, texto, fechas de creación/modificación, autor y estado de publicación.

2. **Multimedia:** Almacena archivos multimedia utilizados en la plataforma. Los atributos proporcionados incluyen información sobre el archivo, como su nombre, tipo, tamaño, URL de almacenamiento, descripción y fecha de carga.

3. **Hipervínculos:** Podría contener enlaces utilizados dentro del contenido, guardando detalles como la URL, el texto del enlace, la fecha de creación y el autor.

4. **Revisiones:** Esta colección esta orientada a revisiones hechas sobre ellos. Almacena detalles sobre la versión, el autor de la revisión y posiblemente los cambios realizados en esa revisión.

5. **Observaciones:** Guarda observaciones realizadas sobre el contenido. Incluye detalles como la fecha de revisión, el autor de la revisión, la observación realizada y posiblemente una categorización del tipo de observación.

6. **Categorías o Secciones:** Puede ser una colección que organiza el contenido en diferentes categorías o secciones. Almacena detalles como el nombre, descripción y etiquetas asociadas a cada categoría o sección.

7. **Comentarios:** Contiene los comentarios hechos sobre el contenido. Guarda el texto del comentario, la fecha de creación y el autor del comentario.

8. **Publicación:** Podría estar relacionada con la programación de publicaciones de contenido. Almacena detalles como la fecha y hora de inicio de publicación y hace referencia al contenido asociado.

9. **Vistas Previa:** Esta colección podría guardar URLs de vistas previas relacionadas con el contenido. Almacena la URL de la vista previa y hace referencia al contenido asociado.

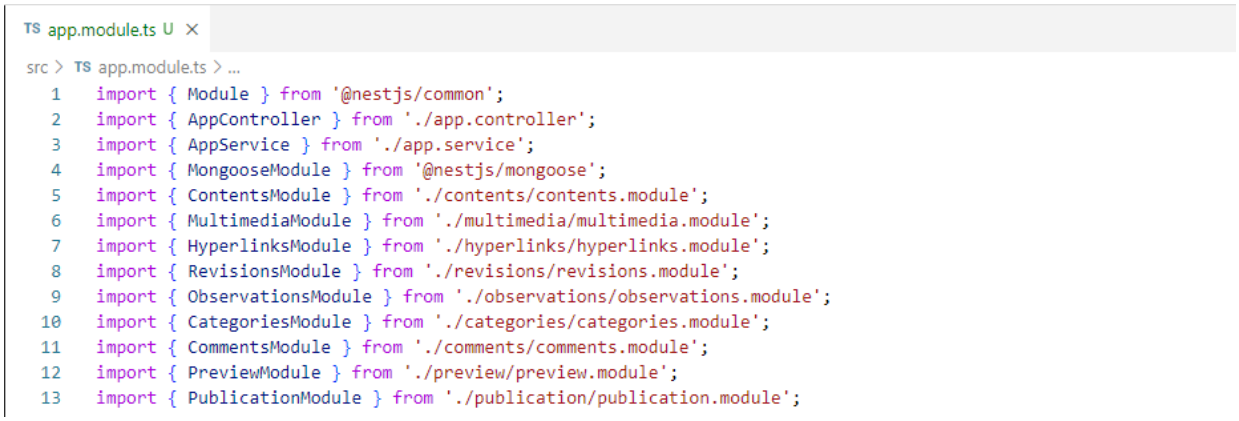
Desarrollo de Endpoints y APIs

Configuración del Entorno

El desarrollo de Creavi se ha estructurado sobre el servidor NestJS, seleccionado como el framework principal para la construcción del backend. Esta elección se fundamenta en la versatilidad y la capacidad de NestJS para la creación de aplicaciones escalables y bien estructuradas.

Uso de Módulos

Se ha adoptado una arquitectura modular para el proyecto, dividiendo el código en módulos independientes para garantizar un diseño organizado y de fácil mantenimiento. Cada módulo encapsula funcionalidades específicas del sistema, promoviendo la reutilización de código y la separación de preocupaciones.



Controladores y Servicios

Cada módulo cuenta con su propio controlador y servicio. Los controladores gestionan las solicitudes HTTP entrantes, definiendo las rutas y manejando las peticiones del cliente. Por otro lado, los servicios encapsulan la lógica de negocio, interactuando con las bases de datos o realizando operaciones específicas para cada entidad.

Ejemplo de controladores y servicios de la colección contents

TS contents.module.ts U

src > contents > TS contents.module.ts > ...

```ts
import { Module } from '@nestjs/common';
import { ContentsController } from './contents.controller';
import { ContentsService } from './contents.service';

@Module({
  controllers: [ContentsController],
  providers: [ContentsService]
})
export class ContentsModule {}
```

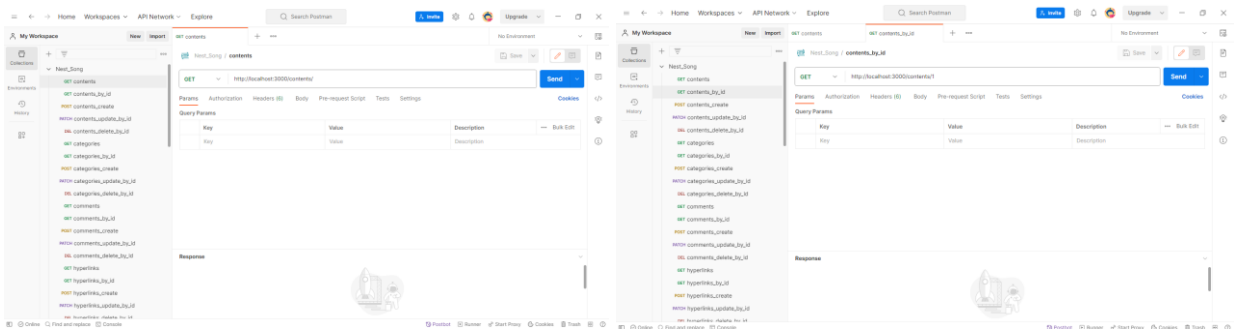
TS contents.service.ts U

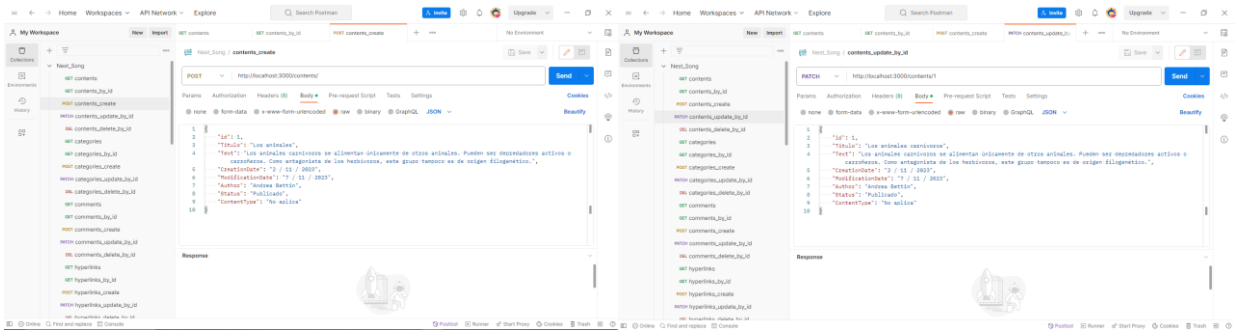
src > contents > TS contents.service.ts > ...

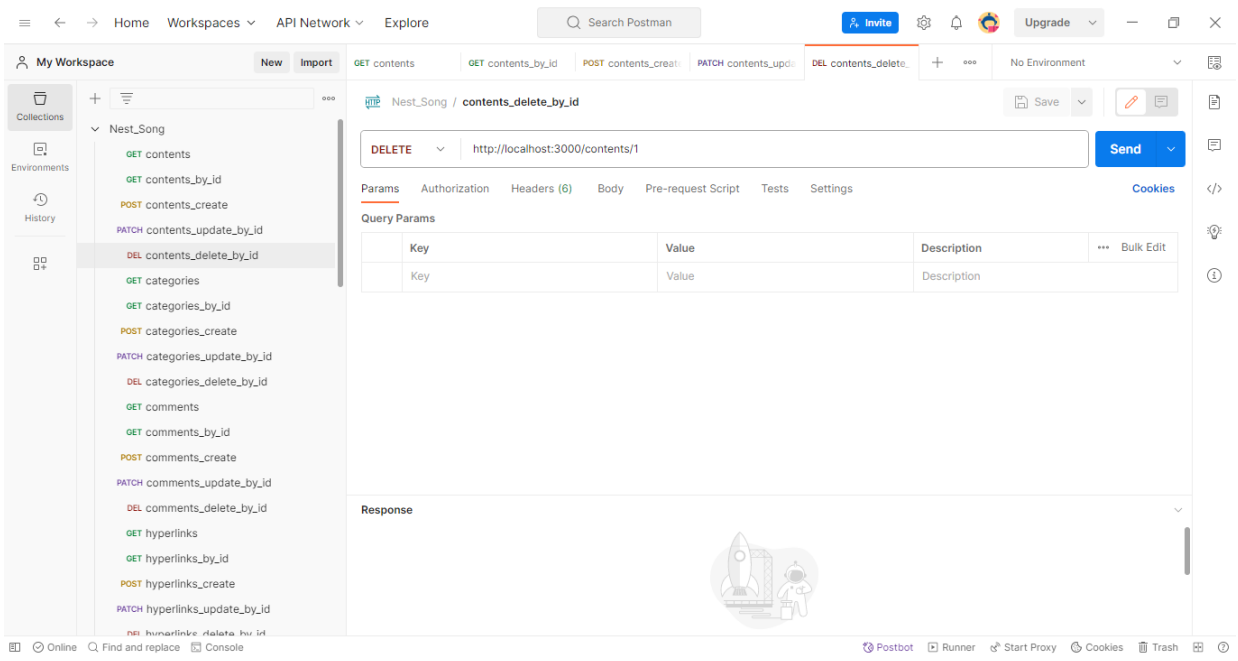
```ts
import { Injectable } from '@nestjs/common';
@Injectable()
export class ContentsService {
private contents: any[] = [];
findOne(id: number): string {
return `Get contents with id ${id}`;
}
findAll(): string {
return 'Get all contentss';
}
create(createContentDto): string {
this.contents.push(createContentDto);
return 'contents created successfully';
}
update(id, updateContent): string {
return 'contents updated successfully';
}
delete(id): string {
return 'contents deleted successfully';
}
}
```

Pruebas de Rutas con Postman

Se llevaron a cabo pruebas exhaustivas de las rutas creadas para cada colección en el desarrollo de Creavi. Estas pruebas se realizaron utilizando Postman, una herramienta de colaboración para el desarrollo de APIs. El objetivo principal fue validar la correcta implementación y funcionamiento de las operaciones CRUD.







Autenticación y Autorización

En este apartado del desarrollo de creavi, el módulo de autenticación y autorización son los encargados del proceso de la autenticación y autorización.

11. Conexión a la Base de Datos

Creación de Base de Datos en MongoDB Atlas

En el marco del desarrollo de Creavi, se ha establecido una sólida base de datos utilizando MongoDB Atlas, una plataforma de gestión de bases de datos en la nube. Se procedió a la creación de una organización específica y, dentro de ella, se configuró un proyecto dedicado al desarrollo de Creavi.

Conexión del Proyecto a MongoDB Atlas

La conexión entre el servidor NestJS y la base de datos en MongoDB Atlas se logró mediante la configuración de las credenciales y la cadena de conexión proporcionadas por MongoDB Atlas. Esto garantiza que el servidor tenga acceso seguro a la base de datos y pueda realizar operaciones de lectura y escritura de manera eficiente.

Organización y Proyecto en MongoDB Atlas

La organización y estructuración del proyecto en MongoDB Atlas permite una gestión efectiva de los recursos de base de datos. La creación de colecciones específicas para cada entidad del sistema asegura una separación clara de datos y facilita la administración y mantenimiento a medida que el proyecto evoluciona.

Configuración de la Conexión

Para conectar Creavi a la base de datos en MongoDB Atlas, se han seguido los siguientes pasos:

Cadena de Conexión:

La cadena de conexión proporcionada por MongoDB Atlas se ha configurado en el archivo de configuración de NestJS, estableciendo los parámetros necesarios, como el nombre de usuario, la contraseña y la URL de la base de datos.

app.module.ts 2, U

src > app.module.ts > ...

```ts
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { MongooseModule } from '@nestjs/mongoose';
import { ContentsModule } from './contents/contents.module';
import { MultimediaModule } from './multimedia/multimedia.module';
import { HyperlinksModule } from './hyperlinks/hyperlinks.module';
import { RevisionsModule } from './revisions/revisions.module';
import { ObservationsModule } from './observations/observations.module';
import { CategoriesModule } from './categories/categories.module';
import { CommentsModule } from './comments/comments.module';
import { PreviewModule } from './preview/preview.module';
import { PublicationModule } from './publication/publication.module';

@Module({
  imports: [MongooseModule.forRoot('mongodb+srv://[redacted]@cluster0.h8kkarc.mongodb.net/?retryWrites=true&w=majority'),
  ContentsModule, MultimediaModule, HyperlinksModule, RevisionsModule, ObservationsModule, CategoriesModule, CommentsModule, PreviewModule,
  PublicationModule],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

Credenciales de Acceso:

Se han creado y gestionado credenciales de acceso seguras para garantizar la autenticación correcta con MongoDB Atlas.

Configuración de Módulos:

Los módulos de conexión a la base de datos han sido configurados en NestJS para garantizar una estructura organizada y modular.

Desarrollo de Operaciones CRUD

Para las operaciones CRUD en Creavi, se han implementado las siguientes funcionalidades:

Crear (Create):

Se ha desarrollado la lógica para agregar nuevos registros a las colecciones correspondientes en la base de datos. Esto incluye la validación de datos y la inserción segura.

Leer (Read):

Se han creado consultas que permiten la recuperación de datos específicos basados en parámetros de búsqueda. Esto abarca la implementación de rutas y controladores para manejar solicitudes de lectura.

Actualizar (Update):

Se ha implementado la capacidad de actualizar registros existentes, garantizando la coherencia y la precisión de la información almacenada.

Eliminar (Delete):

Se han desarrollado funciones para eliminar registros según criterios específicos, asegurando una eliminación segura y controlada.

Manejo de Transacciones

12. Pruebas del Backend

Siguiendo con la etapa del backend, se llegó a la parte final donde se verifica que cada una de las operaciones relacionadas con el backend que el flujo de información con la base de datos en MongoDB se ejecute correctamente.

Diseño de Casos de Prueba

Se diseñaron las pruebas para comprobar las funcionalidades creadas en el crud, se probaron en cada una de las entidades los diferentes métodos, como recuperar por id, crear, actualizar, eliminar, donde en la base de datos se guardaban los cambios hechos en las pruebas.

Ejecución de Pruebas Unitarias y de Integración

En las pruebas unitarias se probaron individualmente cada uno de los métodos (get, get id, post, patch, delete) comprobando el funcionamiento de los mismos y la solución de errores si presentaban al momento de su ejecución.

Manejo de Errores y Excepciones

Etapa 3: Consumo de Datos y Desarrollo Frontend

13. Introducción

Bienvenidos a la tercera etapa del desarrollo de "Creavi", nuestro software innovador para la creación de contenido. En esta fase, nos enfocaremos en el "Consumo de Datos y Desarrollo Frontend", dándole vida a la interfaz que utilizarán nuestros usuarios finales.

El objetivo de esta etapa es diseñar y construir un frontend intuitivo, atractivo y funcional que permita a los usuarios maximizar las capacidades de "creavi". Nos dedicaremos a la integración eficiente de los datos de la fase previa, garantizando una experiencia de usuario coherente, ágil y efectiva.

En esta fase se diseñarán cada uno de los elementos HTML con su estructura y propiedades, que puede incluir un sistema de creación de contenidos, tales como párrafos, títulos, selects, checkboxes y otros elementos de HTML5.

En el desarrollo frontend, se diseñan los aspectos clave que mejoran la experiencia e interacción del usuario con el software Creavi. Estos aspectos incluyen:

Interfaz de Usuario (UI):

- ***Diseño Visual***: Organización de elementos en pantalla, selección de colores, tipografías, iconos, imágenes y otros componentes visuales para crear una apariencia atractiva y uniforme.
- ***Componentes UI***: Botones, menús, formularios y cuadros de diálogo que facilitan la interacción del usuario con la aplicación.

Experiencia de Usuario (UX):

- ***Flujo de Navegación***: Diseño de rutas intuitivas y lógicas por las que los usuarios se desplazan dentro de la aplicación.
- ***Interacción***: Respuesta de los elementos interactivos a las acciones del usuario, como clics y toques, ofreciendo retroalimentación clara.
- ***Accesibilidad***: Diseño inclusivo para que la aplicación sea accesible a personas con distintas capacidades.

Estructura y Contenido:

- ***Maquetación***: Distribución del contenido en la página, empleando diseño responsivo para asegurar una buena visualización en distintos dispositivos y tamaños de pantalla.
- ***Tipografía y Estilo***: Elección y uso de estilos de texto que favorezcan la legibilidad y estética del sitio.

Interacción y Animaciones:

- ***Transiciones*** y Efectos: Animaciones que enriquecen la experiencia del usuario, suavizan la navegación y proveen señales visuales.
- ***Microinteracciones***: Detalles animados que reaccionan a la interacción del usuario, como un botón que cambia de color al pasar el cursor.

Propósito de la Etapa

La fase de "Consumo de Datos y Desarrollo Frontend" en el proyecto Creavi tiene como fin desarrollar la interfaz y experiencia de usuario del software, garantizando que sea atractiva, intuitiva y funcional. Los objetivos específicos de esta fase incluyen:

Transformación del Diseño en Realidad: Convertir los diseños y wireframes en una interfaz interactiva y visualmente coherente.

Mejora de la Experiencia de Usuario (UX): Crear flujos de navegación y patrones de interacción que hagan intuitivo y eficiente el uso de "creavi", reduciendo la curva de aprendizaje.

Integración de Datos: Establecer mecanismos para consumir datos del backend mediante APIs y mostrarlos de forma clara y útil en el frontend.

Optimización de Rendimiento: Garantizar que la aplicación sea rápida y responda bien a las acciones del usuario, optimizando los tiempos de carga y el uso de recursos.

Aseguramiento de la Accesibilidad: Diseñar interfaces accesibles para usuarios con distintas capacidades, siguiendo estándares de accesibilidad web.

Retroalimentación Visual: Implementar animaciones, transiciones y microinteracciones que incrementen la usabilidad y ofrezcan una retroalimentación visual efectiva.

Consistencia Asegurada: Proveer una experiencia uniforme en distintos dispositivos y tamaños de pantalla con un diseño responsivo.

Pruebas y Corrección de Errores: Ejecutar pruebas exhaustivas de la interfaz y corregir errores para asegurar un funcionamiento adecuado y una experiencia de usuario sin inconvenientes.

Alcance de la Etapa

El alcance de la fase de "Consumo de Datos y Desarrollo Frontend" para el software Creavi abarca varias actividades y entregables específicos que se detallan a continuación:

Diseño de la Interfaz de Usuario (UI):

- ***Implementación de Maquetas***: Convertir los wireframes y diseños visuales en código HTML/CSS/JavaScript.
- ***Desarrollo de Componentes***: Crear componentes reutilizables (botones, formularios, menús, etc.) siguiendo los principios de diseño acordados.

Experiencia de Usuario (UX):

- ***Definición de Flujos de Usuario***: Mapear y optimizar los flujos de usuario para diferentes tareas dentro de "creavi".
- ***Prototipos Interactivos***: Desarrollar prototipos interactivos para realizar pruebas de usabilidad y obtener retroalimentación.

Consumo de Datos:

- ***Integración con APIs***: Implementar llamadas a las APIs del backend para obtener y enviar datos, manejando estados de carga, éxito y error.
- ***Gestión de Estado***: Usar herramientas y librerías de gestión de estado (como Redux, Vuex, Context API, etc.) para manejar el estado de la aplicación.

Desempeño y Optimización:

- ***Optimización de Carga***: Minimizar tiempos de carga inicial y subsecuentes, optimizando imágenes, scripts y estilos.
- ***Optimización del Rendimiento***: Mejorar la eficiencia del código para asegurar un rendimiento rápido y fluido, incluyendo el lazy loading de componentes y datos.

Accesibilidad:

- ***Cumplimiento de Estándares***: Asegurar que el frontend cumpla con los estándares de accesibilidad (como WCAG) para ser usable por personas con diversas capacidades.
- ***Pruebas de Accesibilidad***: Realizar pruebas para identificar y corregir problemas de accesibilidad.

Interacciones y Animaciones:

- ***Transiciones y Efectos***: Implementar animaciones y transiciones para mejorar la experiencia de usuario, sin comprometer el rendimiento.
- ***Microinteracciones***: Añadir pequeñas animaciones que mejoren la retroalimentación del usuario (como efectos de hover, clic, etc.).

Testing y Calidad:

- ***Pruebas Unitarias y de Integración***: Escribir y ejecutar pruebas unitarias y de integración para asegurar la estabilidad del frontend.
- ***Pruebas de Usabilidad***: Realizar pruebas de usabilidad con usuarios reales para identificar y solucionar problemas de diseño e interacción.

Documentación:

- ***Documentación del Código***: Crear y mantener documentación clara y detallada del código y componentes desarrollados.
- ***Guías de Uso***: Elaborar guías y manuales para usuarios y desarrolladores sobre cómo usar e implementar el frontend de "creavi".

Definiciones y Acrónimos

Estas son algunas definiciones y acrónimos relevantes para la fase de "Consumo de Datos y Desarrollo Frontend":

UI (Interfaz de Usuario): La interfaz visual a través de la cual los usuarios interactúan con una aplicación. Incluye elementos como botones, formularios, menús y cualquier otro elemento con el que los usuarios puedan interactuar.

UX (Experiencia de Usuario): La experiencia general que tiene un usuario al interactuar con una aplicación o sitio web. Incluye aspectos como la facilidad de uso, la accesibilidad, la eficiencia y la satisfacción del usuario.

HTML (Hypertext Markup Language): El lenguaje estándar utilizado para crear y diseñar páginas web y aplicaciones web. Se utiliza para estructurar el contenido de una página web mediante el uso de etiquetas y elementos.

CSS (Cascading Style Sheets): Un lenguaje de hojas de estilo utilizado para describir la presentación visual de un documento HTML. Se utiliza para definir el diseño, el formato, los colores y otras características visuales de una página web.

JavaScript: Un lenguaje de programación de alto nivel utilizado para agregar interactividad y dinamismo a las páginas web. Se utiliza para manipular el contenido de una página, responder a eventos del usuario y comunicarse con el servidor.

API (Interfaz de Programación de Aplicaciones): Un conjunto de reglas y protocolos que permite a diferentes aplicaciones o sistemas comunicarse entre sí. En el contexto del desarrollo frontend, se refiere a las APIs utilizadas para obtener y enviar datos desde y hacia el backend de una aplicación.

UI/UX Design (Diseño de Interfaz y Experiencia de Usuario): El proceso de diseñar la interfaz visual y la experiencia de usuario de una aplicación o sitio web. Incluye la creación de wireframes, maquetas y prototipos, así como la iteración y mejora continua del diseño basada en la retroalimentación del usuario.

SPA (Single Page Application): Una aplicación web que carga una sola página HTML y actualiza dinámicamente el contenido de esa página en respuesta a las acciones del usuario, sin necesidad de recargar la página completa desde el servidor.

CMS (Sistema de Gestión de Contenidos): Una aplicación o plataforma que permite crear, editar, gestionar y publicar contenido en línea. Algunos ejemplos populares de CMS incluyen WordPress, Drupal y Joomla.

SEO (Optimización de Motores de Búsqueda): El proceso de mejorar la visibilidad de un sitio web en los resultados de búsqueda orgánica de motores de búsqueda. Se centra en aumentar la calidad y la cantidad de tráfico a un sitio web mediante técnicas como la optimización de palabras clave, la creación de enlaces y la mejora del contenido.

14. Creación de la Interfaz de Usuario (UI)

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

En la fase de desarrollo frontend de "creavi", hemos diseñado la interfaz de usuario utilizando Vue.js, un framework progresivo de JavaScript, junto con Vuetify, un framework de componentes UI construido sobre Vue.js. Este enfoque nos ha permitido crear una interfaz de usuario dinámica, atractiva y altamente funcional que cumple con los estándares modernos de diseño web.

Integración con Vue.js

Componentes Vue.js:

- Utilizamos la arquitectura de componentes de Vue.js para dividir la interfaz de usuario en componentes reutilizables y modulares.
- Definimos componentes para secciones específicas de la interfaz, como la barra de navegación, el pie de página, los menús y los formularios.

Gestión del Estado:

- Aprovechamos las capacidades de gestión del estado de Vue.js, específicamente Vuex, para manejar el estado de la aplicación de manera eficiente y predecible.
- Definimos y gestionamos el estado de los componentes para mantener la coherencia de la interfaz de usuario en respuesta a las interacciones del usuario.

Uso de Vuetify

Componentes de Vuetify:

- Utilizamos los componentes predefinidos de Vuetify para construir rápidamente una interfaz de usuario coherente y atractiva.
- Aprovechamos los componentes como v-btn, v-card, v-dialog, v-menu, v-form y otros para crear elementos de interfaz comunes de manera sencilla.

Estilo y Temas:

- Personalizamos el aspecto visual de la interfaz utilizando las opciones de estilo y temas proporcionadas por Vuetify.

- Definimos y aplicamos temas personalizados para asegurar que la interfaz se alinee con la identidad visual de "creavi".

Proceso de Desarrollo

Instalación y Configuración:

- Instalamos Vue.js y Vuetify en nuestro proyecto y los configuramos adecuadamente para comenzar a trabajar.

Desarrollo de Componentes:

- Utilizamos Vue.js para definir componentes reutilizables que encapsulan la lógica y la presentación de partes específicas de la interfaz.
- Utilizamos los componentes de Vuetify para construir interfaces de usuario consistentes y atractivas.

Estilización y Personalización:

- Utilizamos CSS y las opciones de personalización de Vuetify para ajustar el estilo visual de los componentes según sea necesario.

Pruebas y Optimización:

- Realizamos pruebas para garantizar que la interfaz de usuario funcione correctamente en diferentes dispositivos y navegadores.
- Optimizamos el rendimiento y la accesibilidad de la interfaz de usuario según las mejores prácticas de Vue.js y Vuetify.

Consideraciones de Usabilidad

Durante la fase de "Consumo de Datos y Desarrollo Frontend" de "creavi", hemos dado una alta prioridad a la usabilidad de la interfaz de usuario para garantizar una experiencia fluida y satisfactoria para nuestros usuarios finales. Para lograr esto, hemos integrado una serie de consideraciones clave en el diseño y desarrollo de la interfaz de usuario con Vue.js y Vuetify:

Navegación Intuitiva: Hemos diseñado una estructura de navegación clara y coherente que permite a los usuarios moverse fácilmente por la aplicación, facilitando así la exploración y la interacción.

Feedback Visual: Proporcionamos retroalimentación visual clara y oportuna en respuesta a las acciones del usuario, utilizando animaciones y efectos visuales para indicar cambios de estado y confirmaciones de acciones.

Formularios Intuitivos: Los formularios en "creavi" se han diseñado de manera clara y concisa, con etiquetas descriptivas y validación en tiempo real para ayudar a los usuarios a completarlos con precisión y eficiencia.

Accesibilidad: Hemos asegurado que la aplicación cumpla con los estándares de accesibilidad web, utilizando atributos de accesibilidad adecuados y proporcionando alternativas para contenido no textual.

Consistencia Visual y de Interacción: Mantenemos una apariencia visual coherente en toda la aplicación, utilizando temas y estilos consistentes proporcionados por Vuetify, y establecemos patrones de interacción consistentes para acciones comunes.

Carga Rápida y Rendimiento Optimizado: La aplicación se ha optimizado para un tiempo de carga rápido y un rendimiento fluido, minimizando el uso de recursos y aplicando técnicas de optimización frontend.

Pruebas de Usabilidad: Se han realizado pruebas exhaustivas de usabilidad con usuarios reales para identificar y solucionar problemas de diseño e interacción, y se continúa recopilando feedback para iterar y mejorar continuamente la usabilidad de la aplicación.

Maquetación Responsiva

La maquetación responsiva es un componente crucial en el diseño de interfaces de usuario en "creavi". Esto implica que nuestra aplicación se ajusta para proporcionar una experiencia óptima en una diversidad de dispositivos y tamaños de pantalla, desde ordenadores hasta tabletas y móviles.

Principios clave:

Flexibilidad de Diseño: Nuestro diseño se adapta fluida y flexiblemente a distintas resoluciones y orientaciones de pantalla, asegurando una experiencia de usuario uniforme en todos los dispositivos.

Reorganización de Contenido: Los elementos de la interfaz se reorganizan y ajustan dinámicamente para aprovechar al máximo el espacio en pantallas menores, dando prioridad al contenido más relevante y simplificando la navegación.

Media Queries: Empleamos media queries en nuestro CSS para asignar estilos específicos basados en el tamaño y resolución de la pantalla, asegurando una presentación visual idónea en cualquier situación.

Rejillas Flexibles: Utilizamos rejillas flexibles y sistemas de diseño escalables que permiten una distribución proporcional y eficaz del contenido en distintos dispositivos y resoluciones.

Proceso de Implementación:

Análisis de Dispositivos: Determinamos los dispositivos más usados por nuestros usuarios y estudiamos sus características de pantalla para ajustar nuestro diseño efectivamente.

Diseño Basado en Componentes: Creamos componentes y elementos de interfaz modulares y flexibles que pueden reorganizarse con facilidad para distintos tamaños de pantalla.

Pruebas Multiplataforma: Conducimos pruebas exhaustivas en diversos dispositivos y navegadores para asegurar que nuestra aplicación funcione y se visualice correctamente en todas las plataformas.

Optimización de Rendimiento: Nos aseguramos de que nuestra maquetación responsiva sea no solo atractiva visualmente, sino también rápida y eficiente, optimizando el rendimiento y minimizando el tiempo de carga en todos los dispositivos.

15. Programación Frontend con JavaScript (JS)

Desarrollo de la Lógica del Frontend

Manejo de Eventos y Comportamientos Dinámicos

Uso de Bibliotecas y Frameworks (si aplicable)

16. Consumo de Datos desde el Backend

Configuración de Conexiones al Backend

Obtención y Presentación de Datos

Actualización en Tiempo Real (si aplicable)

17. Interacción Usuario-Interfaz

Manejo de Formularios y Validación de Datos

Implementación de Funcionalidades Interactivas

Mejoras en la Experiencia del Usuario

18. Pruebas y Depuración del Frontend

Diseño de Casos de Prueba de Frontend

Pruebas de Usabilidad

Depuración de Errores y Optimización del Código

19. 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

20. Integración con el Backend

Verificación de la Comunicación Efectiva con el Backend

Pruebas de Integración Frontend-Backend

ANEXOS

Diagramas UML

- **Diagrama de Casos de Uso (Use Case Diagram):** Este diagrama muestra las interacciones entre los actores (usuarios) y el sistema. Puede ayudar a identificar las funcionalidades clave y los actores involucrados.
- **Diagrama de Secuencia (Sequence Diagram):** Estos diagramas muestran la interacción entre objetos y actores a lo largo del tiempo. Puedes utilizarlos para representar cómo los usuarios interactúan con la pizarra en un flujo de trabajo específico.

- **Diagrama de Clases (Class Diagram):** Puedes utilizar este diagrama para modelar las clases y estructuras de datos subyacentes en el sistema, como usuarios, pizarras, comentarios, revisiones, etc.
- **Diagrama de Estados (State Diagram):** Este diagrama puede ser útil para modelar el comportamiento de la pizarra en diferentes estados, como "edición", "visualización", "comentario", etc.
- **Diagrama de Despliegue (Deployment Diagram):** Puedes utilizar este diagrama para representar cómo se despliega la aplicación en servidores y cómo interactúa con otros componentes del sistema, como el CMS.
- **Diagrama de Componentes (Component Diagram):** Este diagrama puede ayudar a representar la estructura de componentes del software, como la interfaz de usuario, la lógica de negocio, las bibliotecas y los servicios utilizados.
- **Diagrama de Actividad (Activity Diagram):** Puedes usar este diagrama para modelar flujos de trabajo o procesos específicos, como el flujo de trabajo de creación y edición de contenido en la pizarra.
- **Diagrama de Comunicación (Communication Diagram):** Similar a los diagramas de secuencia, estos diagramas muestran interacciones entre objetos y actores, pero pueden ser más simples y enfocados en la comunicación.
- **Diagrama de Paquetes (Package Diagram):** Este diagrama puede ayudar a organizar y visualizar los paquetes y módulos del software, lo que es útil para el diseño modular.
- **Diagrama de Objetos (Object Diagram):** Puedes utilizar este diagrama para representar instancias de clases y cómo interactúan en un escenario específico.