Codex

DOCUMENTO DE PROPUESTA DE DISEÑO DE SOFTWARE EDUCATIVO I

CREACIÓN DE OBJETO VIRTUAL DE APRENDIZAJE – OVA PARA LA CONCEPTUALIZACIÓN DEL USO DE GIT Y GITHUB.

INTEGRANTES:

Mauro Andrés Monterroza Sevilla
Alexander Domínguez Niño
Maria Claudia Oquendo Méndez
Isacar Torreglosa Díaz
German David Rivera Rosario

TUTOR:
Alexander Toscano Ricardo
@kikret @atoscano

REPOSITORIO:
https://github.com/area-de-informatica/ds1_pa_codex.git

Codex

Creación de objeto virtual de aprendizaje-ova para la conceptualización del uso de git y github

Autores

Mauro Andrés Monterroza Sevilla

mmonterrozasevilla@correo.unicordoba.edu.co

Alexander Domínguez Niño

adomingueznino@correo.unicordoba.edu.co

Maria Claudia Oquendo Méndez

moquendomendez@correo.unicordoba.edu.co

Isacar Torreglosa Díaz

itorreglosadiaz@correo.unicordoba.edu.co

German David Rivera Rosario

Griverarosario73@correo.unicordoba.edu.co

Tutor

Alexander Toscano Ricardo

atoscano@correo.unicordoba.edu.co

Repositorio

https://github.com/area-de-informatica/ds1_pa_codex.git

Descripción del Software

Se propone el desarrollo de un Objeto Virtual de Aprendizaje (OVA) orientado a la enseñanza de Git y GitHub, brindando a los usuarios una experiencia educativa estructurada y dinámica. Este software educativo contará con objetivos claros, contenido didáctico, actividades prácticas y evaluaciones que permitirán reforzar el aprendizaje de conceptos fundamentales como control de versiones, gestión de repositorios y colaboración en proyectos.

El OVA estará diseñado para ser sostenible, escalable y reutilizable en el tiempo, facilitando su adaptación a diferentes contextos educativos. Su estructura modular permitirá la incorporación de nuevos contenidos o actualizaciones sin afectar su funcionamiento general. Además, se priorizará una interfaz intuitiva y accesible, garantizando una experiencia de aprendizaje eficiente e interactiva.

ETAPA 1 DISEÑO DE LA APLICACIÓN Y ANÁLISIS DE REQUISITOS 7

1. INTRODUCCIÓN 7

Propósito del Documento 7
Alcance del Proyecto Módulo de Pizarra Compartida 9
Definiciones y Acrónimos 11

2. DESCRIPCIÓN GENERAL 14

Objetivos del Sistema 14
Funcionalidad General 14
Usuarios del Sistema 15
Restricciones 16

3. REQUISITOS FUNCIONALES 16

Casos de Uso 17
Diagramas de Flujo de Casos de Uso 18
Descripción detallada de cada caso de uso 19
Prioridad de Requerimientos 20

4. REQUISITOS NO FUNCIONALES 22

Requisitos de Desempeño 22
Requisitos de Seguridad 23
Requisitos de Usabilidad 24
Requisitos de Escalabilidad 24

5. MODELADO E/R 25

Diagrama de Entidad-Relación 25
Diagrama Relacional 26
Script de modelo relacional 27
Descripción de Entidades y Relaciones 28
Reglas de Integridad Referencial 30
Colecciones (NoSLQ) 32

6. ANEXOS 34

Diagramas Adicionales 34
Referencias 34

ETAPA 2: PERSISTENCIA DE DATOS CON BACKEND 35

7. INTRODUCCIÓN 35

Propósito de la Etapa 35
Alcance de la Etapa 35
Definiciones y Acrónimos 35

8. DISEÑO DE LA ARQUITECTURA DE BACKEND 35

Descripción de la Arquitectura Propuesta 35
Componentes del Backend 35
Diagramas de Arquitectura 35
9. ELECCIÓN DE LA BASE DE DATOS 35
Evaluación de Opciones (SQL o NoSQL) 35
Justificación de la Elección 35
Diseño de Esquema de Base de Datos 35
10. IMPLEMENTACIÓN DEL BACKEND 35
Elección del Lenguaje de Programación 35
Creación de la Lógica de Negocio 36
Desarrollo de Endpoints y APIs 36
Autenticación y Autorización 36
11. CONEXIÓN A LA BASE DE DATOS 36
Configuración de la Conexión 36
Desarrollo de Operaciones CRUD 36
Manejo de Transacciones 36
12. PRUEBAS DEL BACKEND 36
Diseño de Casos de Prueba 36
Ejecución de Pruebas Unitarias y de Integración 36
Manejo de Errores y Excepciones 36
ETAPA 3: CONSUMO DE DATOS Y DESARROLLO FRONTEND 37
13. INTRODUCCIÓN 37
Propósito de la Etapa 37
Alcance de la Etapa 37
Definiciones y Acrónimos 37
14. CREACIÓN DE LA INTERFAZ DE USUARIO (UI) 37
Diseño de la Interfaz de Usuario (UI) con HTML y CSS 37
Consideraciones de Usabilidad 37
Maquetación Responsiva 37
15. PROGRAMACIÓN FRONTEND CON JAVASCRIPT (JS) 37
Desarrollo de la Lógica del Frontend 37
Manejo de Eventos y Comportamientos Dinámicos 37
Uso de Bibliotecas y Frameworks (si aplicable) 37
16. CONSUMO DE DATOS DESDE EL BACKEND 37
Configuración de Conexiones al Backend 37
Obtención y Presentación de Datos 37

Actualización en Tiempo Real (si aplicable) 38
17. INTERACCIÓN USUARIO-INTERFAZ 38
Manejo de Formularios y Validación de Datos 38
Implementación de Funcionalidades Interactivas 38
Mejoras en la Experiencia del Usuario 38
18. PRUEBAS Y DEPURACIÓN DEL FRONTEND 38
Diseño de Casos de Prueba de Frontend 38
Pruebas de Usabilidad 38
Depuración de Errores y Optimización del Código 38
19. IMPLEMENTACIÓN DE LA LÓGICA DE NEGOCIO EN EL FRONTEND 38
Migración de la Lógica de Negocio desde el Backend (si necesario) 38
Validación de Datos y Reglas de Negocio en el Frontend 38
20. INTEGRACIÓN CON EL BACKEND 38
Verificación de la Comunicación Efectiva con el Backend 38
Pruebas de Integración Frontend-Backend 38
ANEXOS 39

Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

Introducción

Propósito del Documento

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

- **Etapa 1 Diseño de la Aplicación y Análisis de Requisitos**

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

- **Etapa 2: Persistencia de Datos con Backend – Servidor**

En la etapa 2 se continua con los lineamientos de la etapa 1, para seguir adicionando elementos de diseño e implementación de software, enfocados en el desarrollo de APIs, servidores o microservicios que permitan soportar aplicaciones cliente del software educativo; en este sentido, el curso presenta los conceptos de los sistemas de bases de datos, su diseño lógico, la organización de los sistemas manejadores de bases de datos, los lenguaje de definición de datos y el lenguaje de manipulación de datos SQL y NoSQL; de tal manera que los estudiantes adquieran las competencias para analizar, diseñar y desarrollar aplicaciones para gestionar y almacenar grandes cantidades de datos, mediante el uso de técnicas adecuadas como el diseño y modelo lógico y físico de base datos, manejo de los sistemas de gestión de bases de datos, algebra relacional, dominio del lenguaje SQL como herramienta de consulta, tecnología cliente / servidor; igualmente, se definirán los elementos

necesarios para el acceso a dichas bases de datos, como la creación del servidor API, utilizando tecnologías de vanguardia como node.js, express, Nest.js, Spring entre otros; para, finalmente converger en el despliegue de la API utilizando servicios de hospedaje en la nube, preferiblemente gratuitos. También podrá implementar servidores o API's con inteligencia artificial o en su defecto crear una nueva capa que consuma y transforme los datos obtenidos de la IA.

El desarrollo del curso se trabajara por proyectos de trabajo colaborativo que serán evaluados de múltiples maneras, teniendo en cuenta más el proceso que el resultado.

- **Etapa 3: Consumo de Datos y Desarrollo Frontend – Cliente**

La etapa 3 el estudiante está en capacidad de establecer la mejor elección de herramientas de consumo de datos y técnicas en aras de lograr el mejor producto a nivel de software o hardware acorde a los requerimientos funcionales y no funcionales del problema a solucionar. En este punto el estudiante puede consumir los datos a través de un cliente que puede ser una aplicación de celular, una aplicación de escritorio, una página web, IoT(internet de las cosas) o incluso, artefactos tecnológicos.

El diseño gráfico es de los requisitos esenciales en la capa de presentación, por lo tanto, se requieren los cursos de diseño gráfico vistos previamente. Los elementos anteriores nos permiten elegir el paradigma y tecnología para desarrollar nuestras aplicaciones, teniendo en cuenta que podríamos desarrollar aplicaciones de tipo cliente.

Análisis de requisitos:

1. Requisitos Funcionales

1.1. Contenidos Educativos

- Módulos estructurados para la enseñanza de Git y GitHub.
- Material didáctico en diversos formatos (videos, textos explicativos, ejemplos prácticos).
- Actividades interactivas para aplicar conceptos clave.
- Evaluaciones automáticas al finalizar cada módulo con retroalimentación inmediata.

1.2. Seguimiento del Aprendizaje

- Indicadores de progreso en cada módulo.
- Retroalimentación detallada sobre errores en las actividades.
- Posibilidad de repetir actividades y mejorar resultados.

1.3. Sostenibilidad y Reutilización

- Estructura modular que permita actualizar o añadir nuevos temas sin modificar el sistema base.
- Compatibilidad con distintos dispositivos y navegadores.
- Facilidad para integrar nuevos ejercicios o adaptarlo a diferentes niveles de aprendizaje.

2. Requisitos No Funcionales

2.1. Usabilidad

- Interfaz intuitiva y clara para facilitar el aprendizaje.
- Diseño visual atractivo con elementos gráficos que refuercen la comprensión.
- Navegación sencilla entre módulos y actividades.

2.2. Rendimiento y Escalabilidad

- Carga rápida de contenidos y ejercicios.
- Funcionamiento fluido sin depender de instalaciones externas.
- Capacidad para incorporar más funcionalidades en el futuro sin afectar el desempeño.

2.3. Seguridad

- Protección de los datos generados por los usuarios en sus actividades.

- Acceso seguro a los recursos sin necesidad de registros complejos.

2.4. Mantenimiento y Actualización

- Documentación clara para futuras modificaciones o mejoras.
- Facilidad para actualizar contenidos educativos y ejercicios prácticos.

Sistema modular que permita añadir nuevas funciones sin afectar la estabilidad del OVA.

Alcance del Proyecto OVA sobre la conceptualización del uso de GIT y GITHUB

El OVA tiene como objetivo desarrollar un entorno educativo digital interactivo y estructurado que permita a los usuarios adquirir competencias teóricas y prácticas en el uso de Git como sistema de control de versiones distribuido y GitHub como plataforma de hospedaje y colaboración de proyectos. Desde una perspectiva técnica, el OVA estará construido bajo una estructura modular y flexible que permitirá su crecimiento y actualización sin afectar su funcionamiento principal, garantizando que pueda adaptarse fácilmente a nuevos contenidos y mejoras futuras.

El OVA presentará los conceptos de manera progresiva y amigable, utilizando recursos como videos breves, textos explicativos claros, infografías didácticas y simulaciones básicas de comandos. Se diseñarán actividades interactivas guiadas paso a paso, con retroalimentación inmediata directamente en el navegador, permitiendo a los estudiantes aprender de forma práctica y segura, sin requerir instalaciones adicionales. También se garantizará que el OVA sea accesible desde computadores, tablets y teléfonos móviles, empleando un diseño responsivo que se adapte a diferentes tamaños de pantalla. Para fomentar el aprendizaje autónomo, el OVA incluirá herramientas de autoevaluación, visualización del progreso, y la posibilidad de repetir actividades las veces que se necesite. A nivel técnico, también se dejarán sentadas las bases para que, en el futuro, se puedan integrar funciones como foros de discusión y modos de estudio offline mediante tecnologías como aplicaciones web progresivas.

Finalmente, se cuidará especialmente que la plataforma cumpla principios de accesibilidad web, usando interfaces intuitivas, colores de alto contraste, y glosarios emergentes que expliquen términos técnicos, facilitando el uso del OVA incluso para estudiantes sin experiencia previa en control de versiones ni en herramientas de desarrollo.

Funcionalidades

- Simular el uso básico de comandos.
- Generar actividades.
- Generar reporte de actividades completadas.
- Mostrar progreso.
- Desplegar explicaciones breves para respuestas incorrectas.
- Mostrar sección de preguntas frecuentes (FAQ).
- Generar contenido en PDF.
- Generar ayudas contextuales.
- Conectar con API.
- Integrar con CMS.
- Integrar con chatbot.

Activar las opciones de accesibilidad:

- Activar modo nocturno.
- Activar Subtítulos.
- Mostrar Controles de Audio.
- Seleccionar Idioma.
- Activar lector de pantalla.

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).

Descripción General

Objetivos del Sistema

El objetivo del sistema es proporcionar una pizarra compartida dentro de un Sistema de Gestión de Contenido llamado CREAVI que permita a los usuarios colaborar de manera eficiente y efectiva, facilitando la creación, visualización y edición de contenido visual en tiempo real. Esta pizarra compartida se diseñará con el propósito de mejorar la comunicación y la colaboración en un entorno en línea, ofreciendo a los usuarios una plataforma intuitiva y versátil para crear y compartir ideas, diagramas, esquemas y contenido visual de manera colaborativa, enriqueciendo así la experiencia de usuario y la productividad en el uso del CMS.

Funcionalidad General

- **Creación y Edición Colaborativa**: Permite a los usuarios crear y editar contenido en la pizarra de forma colaborativa en tiempo real. Múltiples usuarios pueden trabajar en el mismo documento simultáneamente.
- **Herramientas de Dibujo y Anotación**: Proporciona herramientas de dibujo, pinceles, formas y opciones de anotación que permiten a los usuarios plasmar sus ideas y conceptos de manera visual.
- **Carga de Imágenes y Multimedia**: Permite a los usuarios cargar imágenes, videos y otros medios directamente en la pizarra, lo que facilita la ilustración de conceptos.
- **Organización de Contenido**: Ofrece opciones para organizar y estructurar el contenido en la pizarra, como la creación de capas, agrupación de elementos y uso de etiquetas.
- **Historial de Revisiones**: Registra un historial de revisiones que permite a los usuarios rastrear los cambios realizados en la pizarra y restaurar versiones anteriores si es necesario.
- **Compartir y Colaborar**: Permite compartir la pizarra con otros usuarios a través de enlaces o invitaciones, lo que facilita la colaboración con colegas, clientes o amigos.
- **Comentarios y Chat en Tiempo Real**: Los usuarios pueden comentar y discutir sobre el contenido de la pizarra a través de un chat en tiempo real, lo que facilita la comunicación durante la colaboración.

- **Exportación e Impresión**: Ofrece la capacidad de exportar el contenido de la pizarra en varios formatos (PDF, imagen, etc.) y la opción de imprimirlo.
- **Integración con el CMS**: Se integra de manera transparente con el sistema de gestión de contenido (CMS CREAVI), lo que permite incrustar pizarras en los contenidos, metodologías o cualquier otro tipo de componente que permita la pizarra.
- **Personalización y Temas**: Permite a los usuarios personalizar la apariencia de la pizarra y seleccionar temas que se adapten a sus necesidades.
- **Acceso Seguro**: Proporciona medidas de seguridad para garantizar que solo los usuarios autorizados puedan acceder y editar la pizarra.
- **Notificaciones y Actualizaciones en Tiempo Real**: Los usuarios reciben notificaciones sobre cambios en la pizarra y pueden ver actualizaciones en tiempo real mientras otros editan.
- **Acceso Móvil**: Ofrece una experiencia de usuario optimizada en dispositivos móviles, permitiendo el acceso y la colaboración desde smartphones y tabletas.
- **Búsqueda y Filtros**: Facilita la búsqueda de contenido en la pizarra y la aplicación de filtros para organizar y encontrar información específica.
- **Gestión de Usuarios y Permisos**: Permite a los administradores gestionar usuarios y definir permisos de acceso y edición.
- **Informes y Analíticas**: Proporciona información sobre el uso de la pizarra, como quién la ha editado, cuándo se realizaron cambios y estadísticas sobre el contenido(XAPI).

Usuarios del Sistema

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

Funcionalidad	Administradores	Docente Investigador	Docente Invitado	Alumno	Invitado
Creación y Edición		✓	✓	✓	
Herramientas de Dibujo		✓	✓	✓	
Carga de Imágenes y Multimedia		✓	✓	✓	
Edicion de Imágenes y Multimedia		✓	✓	✓	

Organización de Contenido		✓	✓		
Historial de Revisiones	✓	✓	✓		
Compartir y Colaborar	✓	✓	✓		
Comentarios y Chat	✓	✓	✓	✓	✓
Exportación e Impresión		✓	✓	✓	
Integración con el CMS	✓	✓	✓		
Personalización y Temas	✓	✓	✓		
Acceso Seguro	✓	✓	✓	✓	
Notificaciones en Tiempo Real	✓	✓	✓	✓	
Acceso Móvil	✓	✓	✓	✓	✓
Búsqueda y Filtros	✓	✓	✓	✓	✓
Gestión de Usuarios y Permisos	✓				
Informes y Analíticas	✓	✓	✓		

Restricciones

Solo usuarios agregados por un anfitrión de la pizarra tendrán acceso a las funcionalidades descritas en la tabla anterior, un anfitrión puede agregar otros anfitriones a la pizarra quienes pueden ser docentes, alumnos o invitados, también se les puede dar el rol de moderador y/o administrador de pizarra. Las funcionalidades de estos dos roles no se han descrito aún.

Requisitos Funcionales

Creación y Edición de Contenido:

- Los usuarios deben poder crear y editar contenido en tiempo real en la pizarra.
- Los usuarios deben tener acceso a herramientas de dibujo, anotación y edición de contenido.

Compartir y Colaborar:

- Los usuarios deben poder compartir la pizarra con otros usuarios a través de enlaces o invitaciones.
- Múltiples usuarios deben poder colaborar en la misma pizarra simultáneamente.

Carga de Multimedia:

- Los usuarios deben poder cargar imágenes, videos y otros medios directamente en la pizarra.

Historial de Revisiones:

La aplicación debe mantener un historial de revisiones que permita a los usuarios rastrear cambios realizados en la pizarra y restaurar versiones anteriores.

Comentarios y Chat en Tiempo Real:

- Los usuarios deben poder comentar y discutir sobre el contenido de la pizarra a través de un chat en tiempo real.

Exportación e Impresión:

- Los usuarios deben poder exportar el contenido de la pizarra en varios formatos (PDF, imagen, etc.) y tener la opción de imprimirlo.

Integración con el CMS:

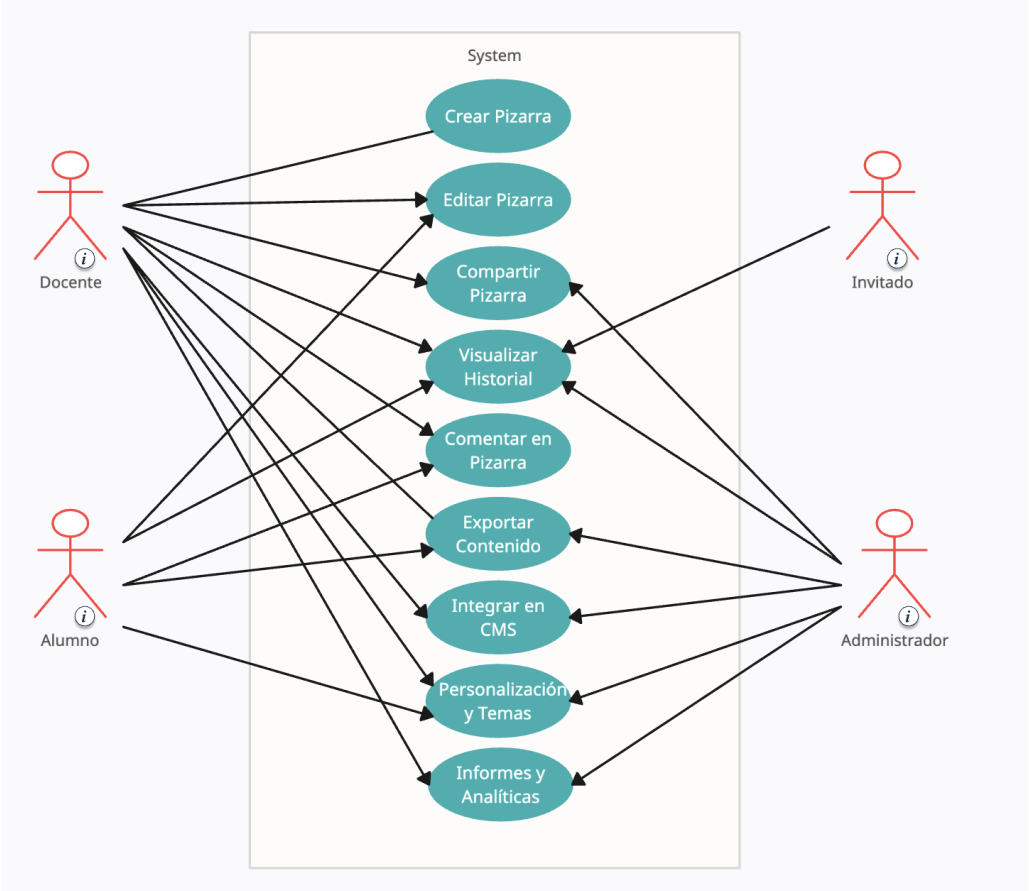
- La pizarra debe integrarse con el sistema de gestión de contenidos (CMS) existente para permitir la incrustación de pizarras en páginas web o artículos.

Personalización y Temas:

- Los usuarios deben poder personalizar la apariencia de la pizarra y seleccionar temas que se adapten a sus necesidades.

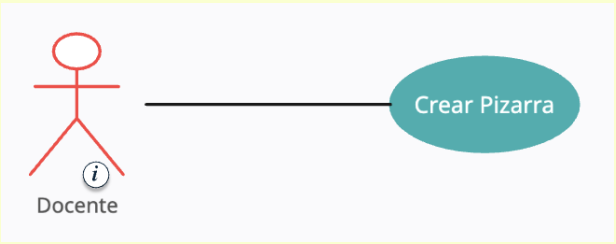
Casos de Uso

Diagrama de caso de uso



https://app.creately.com/d/start/dashboard

Diagramas de Flujo de Casos de Uso



Prioridad: requerido
Versión: 1
Complejidad: media

SP: Seleccionar Plantilla
AA: Agregar Actores
GP:

1. Crear Pizarra

Flujo: Crear pizarra

Prueba: Seleccionar nueva pizarra, elegir tipo de pizarra.

a. Crear pizarra

Flujo: SP, AA, GP

CASO No. 1 Crear Pizarra

ID:	CU-1	
Nombre	Crear Pizarra	
Actores	Docente investigador, Docente invitado	
Objetivo	Este caso debe permitir crear una pizarra	
Urgencia	5	
Esfuerzo	4	
Pre-condiciones	- Debe haberse autenticado de forma correcta en el sistema. - La pizarra se agrega a un contenido o módulo que la admita	
Flujo Normal	Docente	Sistema
	Selecciona nueva pizarra	
		Despliega las plantillas disponibles
	Selecciona una plantilla	
	Selecciona agrega colaboradores	
		Retorna usuarios estudiantes o docentes.
	Agrega colaboradores	
	Registra la plantilla	
		Guarda la pizarra
Flujo alternativo 1		
Flujo alternativo 2		
Post-condiciones		
Exepciones		

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
	4-Alto	4	8	12	16	20
						CU-1
	3-Medio	3	6	9	12	15
	2-Bajo	2	4	6	8	10
	1-Muy bajo	1	2	3	4	5

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

Requisitos No Funcionales

Seguridad:

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

Rendimiento:

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

Escalabilidad:

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

Disponibilidad:

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

Compatibilidad con Dispositivos:

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

Usabilidad:

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

Accesibilidad:

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

Cumplimiento Normativo:

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

Tiempo de Respuesta:

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

Requisitos de Desempeño

1. **Rendimiento en Tiempo Real**: La pizarra debe proporcionar un rendimiento en tiempo real, lo que significa que los cambios realizados por los usuarios deben reflejarse instantáneamente para todos los colaboradores, incluso cuando múltiples usuarios trabajen simultáneamente en la pizarra. Este aspecto se debe desarrollar con sockets.

2. **Tiempo de Carga Rápido**: La pizarra debe cargar de manera eficiente, y los usuarios no deben experimentar tiempos de carga excesivamente largos al acceder a una pizarra o al editar contenido. Se requiere que los componentes estén bien diseñados y acoplados. Por lo general los componentes de la arquitectura CREAVI la cual sigue el paradigma de la programación orientada a componentes.
3. **Optimización de Recursos**: El sistema debe estar optimizado para utilizar eficientemente los recursos del servidor, minimizando el uso de CPU y memoria. El renderizado de los componentes adecuados garantiza este requisito.

Requisitos de Seguridad

4. **Acceso Seguro**: Se debe implementar una autenticación segura para garantizar que solo usuarios autorizados tengan acceso a las pizarras. Esto puede incluir autenticación de dos factores, inicio de sesión único (SSO), autenticación con JWT o Auth 2.
5. **Protección de Datos**: La pizarra debe garantizar la protección de datos sensibles, como información del usuario y contenido compartido. Se debe cifrar la información en tránsito y en reposo.
6. **Auditoría y Registro de Actividades**: El sistema debe mantener registros de actividades, lo que incluye registros de cambios en la pizarra, acceso de usuarios y eventos relevantes para la seguridad.
7. **Control de versiones**: El sistema debe llevar un registro de los cambios de los datos gestionados en la pizarra así como también los datos mismos de la estructura del componente.
8. **Variables de entorno**: El sistema debe ser manejado con variables de entorno que garanticen su fácil incorporación con otros módulos y la migración entre plataformas, así como tambien almacenar los datos iniciales del servidor como lo son las bases de datos y las llaves de autenticación, entre otras.

Requisitos de Usabilidad

9. **Interfaz Intuitiva:** La interfaz de usuario de la pizarra debe ser intuitiva y fácil de usar, permitiendo a los usuarios realizar acciones como dibujar, agregar contenido y colaborar sin dificultad.
10. **Compatibilidad con Dispositivos:** La pizarra debe ser compatible con una variedad de dispositivos, incluyendo computadoras de escritorio, tabletas y dispositivos móviles, y debe adaptarse a diferentes tamaños de pantalla.
11. **Documentación y Ayuda en Línea:** Se debe proporcionar documentación clara y ayuda en línea para los usuarios, incluyendo tutoriales y recursos de soporte.

Requisitos de Escalabilidad

12. **Manejo de Cargas Elevadas:** El sistema debe ser escalable para manejar un gran número de usuarios y múltiples pizarras simultáneamente, sin degradación significativa del rendimiento.
13. **Balanceo de Carga:** Se debe implementar un mecanismo de balanceo de carga para distribuir las solicitudes de usuarios de manera equitativa entre los servidores para garantizar la escalabilidad.
14. **Arquitectura de Backend Escalable:** La arquitectura del backend debe estar diseñada para escalar horizontalmente, lo que permite agregar más recursos de hardware a medida que aumenta la demanda.

Modelado E/R

Diagrama de Entidad-Relación

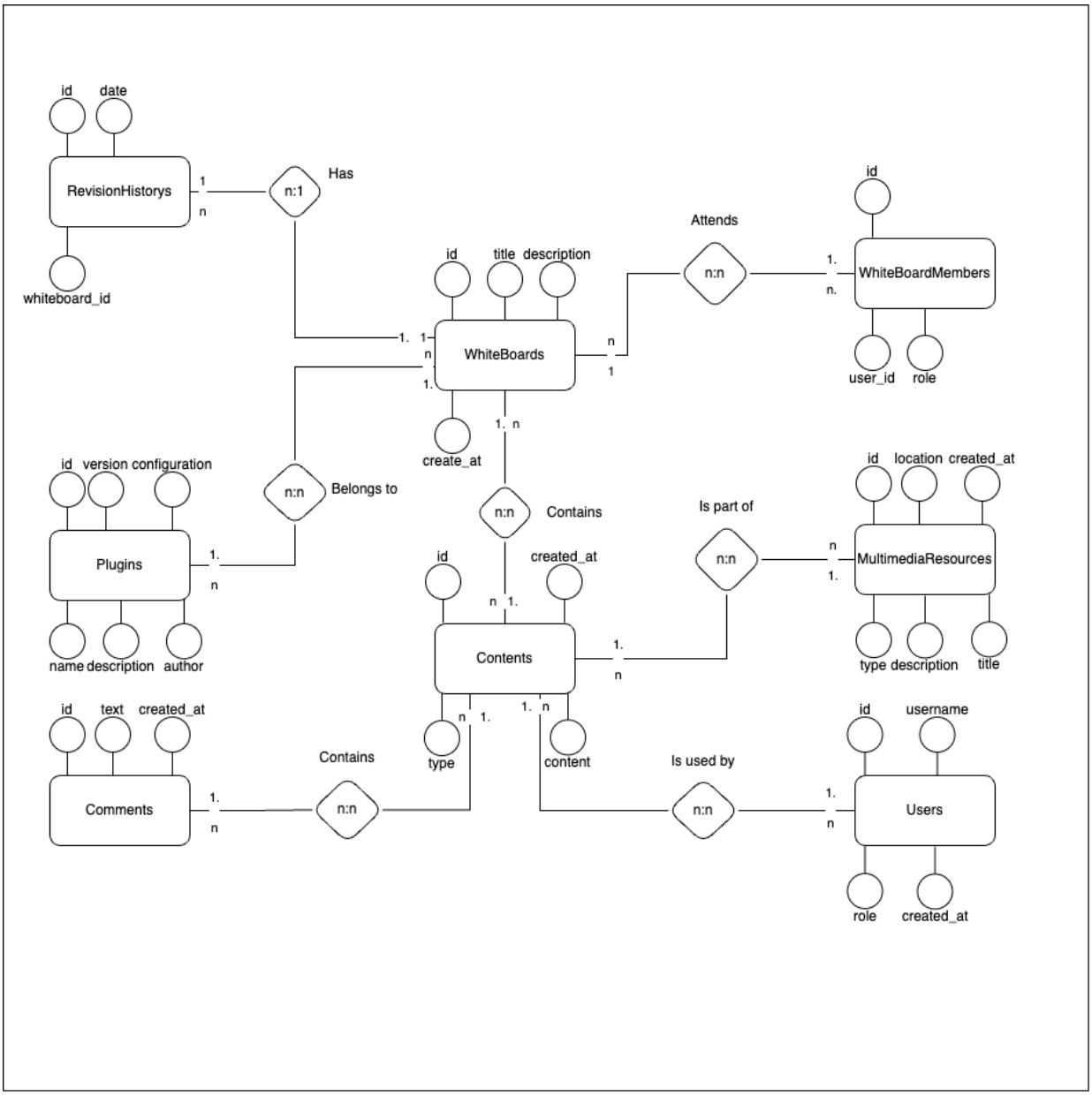
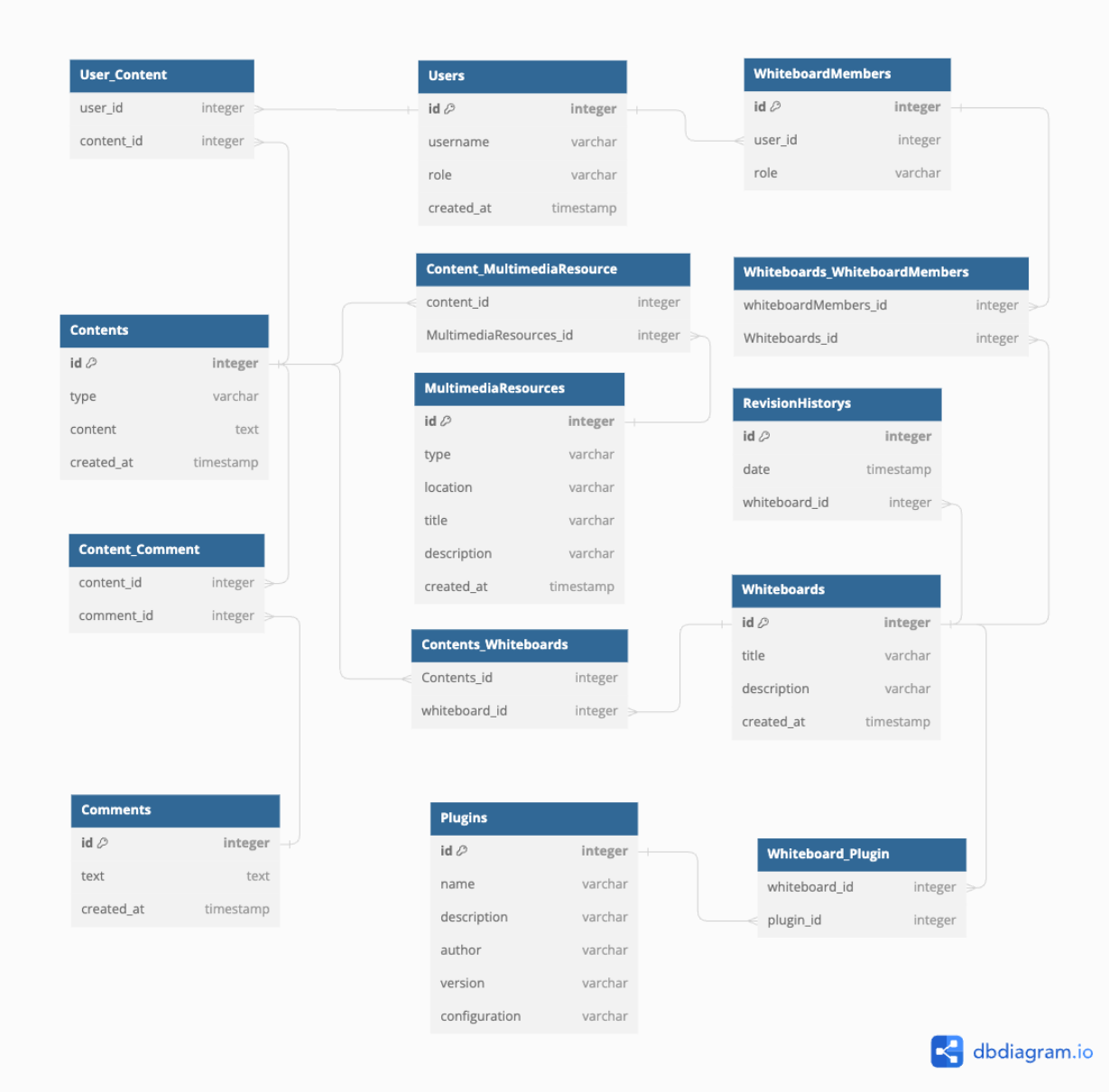


Diagrama Relacional



Script de modelo relacional

https://dbdiagram.io/

```
Table Users {
  id integer [primary key]
  username varchar
  role varchar
  created_at timestamp
}

Table User_Content {
  user_id integer [ref: > Users.id]
  content_id integer [ref: > Contents.id]
}

Table Content_Comment {
  content_id integer [ref: > Contents.id]
  comment_id integer [ref: > Comments.id]
}

Table Whiteboards {
  id integer [primary key]
  title varchar
  description varchar
  created_at timestamp
}

Table Whiteboards_WhiteboardMembers {
  whiteboardMembers_id integer [ref: > WhiteboardMembers.id]
  Whiteboards_id integer [ref: > Whiteboards.id]
}

Table Content_MultimediaResource {
  content_id integer [ref: > Contents.id]
  MultimediaResources_id integer [ref: > MultimediaResources.id]
}

Table Comments {
  id integer [primary key]
  text text
  created_at timestamp
}

Table RevisionHistorys {
  id integer [primary key]
  date timestamp
  whiteboard_id integer [ref: > Whiteboards.id]
}

Table MultimediaResources {
  id integer [primary key]
  type varchar
  location varchar
  title varchar
  description varchar
  created_at timestamp
}

Table Plugins {
  id integer [primary key]
  name varchar
```

```
Table Whiteboard_Plugin {
  whiteboard_id integer [ref: > Whiteboards.id]
  plugin_id integer [ref: > Plugins.id]
}

Table Contents {
  id integer [primary key]
  type varchar
  content text [note: 'Content of the content']
  created_at timestamp
}

Table Contents_Whiteboards {
  Contents_id integer [ref: > Contents.id]
  whiteboard_id integer [ref: > Whiteboards.id]
}
```

```
  description varchar
  author varchar
  version varchar
  configuration varchar
}

Table WhiteboardMembers {
  id integer [primary key]
  user_id integer [ref: > Users.id]
  role varchar
}
```

Descripción de Entidades y Relaciones

Entidades:

1. **User (Usuario):**

   Almacena información sobre los usuarios que pueden acceder a la pizarra.
   Atributos: ID (identificador único), nombre de usuario, rol (como administrador, docente, estudiante), fecha de creación.
   Relaciones: Cada usuario puede estar asociado con varias pizarras a través de la entidad "WhiteboardMember."
2. **Whiteboard (Pizarra):**

   Representa una pizarra en la aplicación de pizarra compartida.
   Atributos: ID (identificador único), título de la pizarra, descripción, fecha de creación.
   Relaciones: Cada pizarra puede contener contenido a través de la relación con la entidad "Content" y puede tener plugins asociados a través de la relación con la entidad "Plugin."
3. **Content (Contenido):**

Almacena contenido que se puede agregar a las pizarras, como texto, imágenes, videos, documentos, etc.
Atributos: ID (identificador único), tipo de contenido, contenido en sí, fecha de creación.
Relaciones: El contenido se asocia con un usuario a través de la relación con la entidad "User" y puede estar relacionado con comentarios.

4. **Comment (Comentario):**

   Almacena comentarios realizados por los usuarios en relación con el contenido de la pizarra.
   Atributos: ID (identificador único), texto del comentario, fecha de creación.
   Relaciones: Cada comentario se relaciona con el contenido específico en la entidad "Content."
5. **RevisionHistory (Historial de Revisiones):**

   Registra el historial de revisiones y cambios realizados en las pizarras.
   Atributos: ID (identificador único), fecha de la revisión.
   Relaciones: Cada entrada de historial se relaciona con una pizarra específica en la entidad "Whiteboard."
6. **MultimediaResource (Recurso Multimedia):**

   Almacena recursos multimedia, como imágenes, videos, documentos, etc.
   Atributos: ID (identificador único), tipo de recurso, ubicación o URL del recurso, título, descripción, fecha de carga.
   Relaciones: Cada entrada de Recurso multimedia se relaciona con una un contenido específico en la entidad " Content."
7. **Plugin:**

   Almacena información sobre los plugins que pueden proporcionar funcionalidades personalizadas en las pizarras.
   Atributos: ID (identificador único), nombre del plugin, descripción, autor, versión, configuración.
   Relaciones: Cada pizarra puede tener asociado uno o varios plugins a través de la entidad "Whiteboard."

Relaciones:

- "User" se relaciona con "WhiteboardMember" para indicar la asociación de los usuarios con las pizarras.
- "Content" se relaciona con "User" para registrar los usuarios que pueden interactuar con el contenido.
- "Content" se relaciona con "Comment" para permitir comentarios en los contenidos.

- "Whiteboard" se relaciona con "Content" para indicar que una pizarra puede contener contenido.
- "Whiteboard" se relaciona con "Plugin" para permitir la asociación de plugins con las pizarras.
- "WhiteboardMember" se relaciona con "User" y "Whiteboard" para indicar la asociación de usuarios con pizarras y sus roles.
- "RevisionHistory" se relaciona con "Whiteboard" para registrar revisiones en las pizarras.
- " MultimediaResource " se relaciona con "Content" para registrar recursos asociados en el contenido.

Reglas de Integridad Referencial

1. **Integridad Referencial entre "User_Content" y "Users"**: Cada registro en la tabla "User_Content" debe estar asociado con un usuario existente en la tabla "Users" a través de la clave foránea "user_id."
2. **Integridad Referencial entre "User_Content" y "Contents"**: Cada registro en la tabla "User_Content" debe estar asociado con un contenido existente en la tabla "Contents" a través de la clave foránea "content_id."
3. **Integridad Referencial entre "Content_Comment" y "Contents"**: Cada registro en la tabla "Content_Comment" debe estar asociado con un contenido existente en la tabla "Contents" a través de la clave foránea "content_id."
4. **Integridad Referencial entre "Content_Comment" y "Comments"**: Cada registro en la tabla "Content_Comment" debe estar asociado con un comentario existente en la tabla "Comments" a través de la clave foránea "comment_id."
5. **Integridad Referencial entre "Whiteboards_WhiteboardMembers" y "Whiteboards"**: Cada registro en la tabla "Whiteboards_WhiteboardMembers" debe estar asociado con una pizarra existente en la tabla "Whiteboards" a través de la clave foránea "Whiteboards_id."
6. **Integridad Referencial entre "Whiteboards_WhiteboardMembers" y "WhiteboardMembers"**: Cada registro en la tabla "Whiteboards_WhiteboardMembers" debe estar asociado con un miembro de la pizarra existente en la tabla "WhiteboardMembers" a través de la clave foránea "whiteboardMembers_id."

7. **Integridad Referencial entre "Whiteboard_Plugin" y "Whiteboards"**: Cada registro en la tabla "Whiteboard_Plugin" debe estar asociado con una pizarra existente en la tabla "Whiteboards" a través de la clave foránea "whiteboard_id."
8. **Integridad Referencial entre "Whiteboard_Plugin" y "Plugins"**: Cada registro en la tabla "Whiteboard_Plugin" debe estar asociado con un plugin existente en la tabla "Plugins" a través de la clave foránea "plugin_id."
9. **Integridad Referencial entre "Contents_Whiteboards" y "Contents"**: Cada registro en la tabla "Contents_Whiteboards" debe estar asociado con un contenido existente en la tabla "Contents" a través de la clave foránea "Contents_id."
10. **Integridad Referencial entre "Contents_Whiteboards" y "Whiteboards"**: Cada registro en la tabla "Contents_Whiteboards" debe estar asociado con una pizarra existente en la tabla "Whiteboards" a través de la clave foránea "whiteboard_id."
11. **Integridad Referencial entre "Content_MultimediaResource" y "Contents"**: Cada registro en la tabla "Content_MultimediaResource" debe estar asociado con un contenido existente en la tabla "Contents" a través de la clave foránea "content_id."
12. **Integridad Referencial entre "Content_MultimediaResource" y "MultimediaResources"**: Cada registro en la tabla "Content_MultimediaResource" debe estar asociado con un recurso multimedia existente en la tabla "MultimediaResources" a través de la clave foránea "MultimediaResources_id."
13. **Integridad Referencial entre "RevisionHistorys" y "Whiteboards"**: Cada registro en la tabla "RevisionHistorys" debe estar asociado con una pizarra existente en la tabla "Whiteboards" a través de la clave foránea "whiteboard_id."
14. **Integridad Referencial entre "WhiteboardMembers" y "Users"**: Cada registro en la tabla "WhiteboardMembers" debe estar asociado con un usuario existente en la tabla "Users" a través de la clave foránea "user_id."

Colecciones (NoSLQ)

Users: {

  _id: ObjectId,

  username: String,

  role: String,

  created_at: date,

  contents_id: [ObjectId],

}

Whiteboards: {

  _id: ObjectId,

  title: String,

  description: String,

  created_at: Date,

  contents_id: [ObjectId],

  whiteboardMembers_id: [ObjectId],

  plugin_id: [ObjectId],

}

Contents: {

  _id: ObjectId,

  type: String,

  content: String,

  created_at: Date,

  user_id: [ObjectId],

  comment_id: [ObjectId],

  whiteboard_id: [ObjectId],

  multimediaResources_id: [ObjectId],

}

WhiteboardMembers: {

  _id: ObjectId,

  user_id: ObjectId,

Comments: {

  _id: ObjectId,

  text: String,

  created_at: ObjectId,

  contents_id: [ObjectId],

}

RevisionHistorys: {

  _id: ObjectId,

  Date: Date,

  whiteboard_id: [ObjectId],

}

MultimediaResources: {

  _id: ObjectId,

  Type: String,

  Location: String,

  Title: String,

  Description: String,

  created_at: Date,

  content_id: [ObjectId],

}

Plugins: {

  _id: ObjectId,

  Name: String,

  Description: String,

  Author: ObjectId,

  Version: String,

  Configuration: Object,

  whiteboard_id: [ObjectId],

}

```
  role: String,

  whiteboards_id: [ObjectId],

}
```

Anexos

Diagramas Adicionales

Referencias

Etapa 2: Persistencia de Datos con Backend

Introducción

Propósito de la Etapa

Alcance de la Etapa

Definiciones y Acrónimos

Diseño de la Arquitectura de Backend

Descripción de la Arquitectura Propuesta

Componentes del Backend

Diagramas de Arquitectura

Elección de la Base de Datos

Evaluación de Opciones (SQL o NoSQL)

Justificación de la Elección

Diseño de Esquema de Base de Datos

Implementación del Backend

Elección del Lenguaje de Programación

Creación de la Lógica de Negocio

Desarrollo de Endpoints y APIs

Autenticación y Autorización

Conexión a la Base de Datos

Configuración de la Conexión

Desarrollo de Operaciones CRUD

Manejo de Transacciones

Pruebas del Backend

Diseño de Casos de Prueba

Ejecución de Pruebas Unitarias y de Integración

Manejo de Errores y Excepciones

Etapa 3: Consumo de Datos y Desarrollo Frontend

Introducción

Propósito de la Etapa

Alcance de la Etapa

Definiciones y Acrónimos

Creación de la Interfaz de Usuario (UI)

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

Consideraciones de Usabilidad

Maquetación Responsiva

Programación Frontend con JavaScript (JS)

Desarrollo de la Lógica del Frontend

Manejo de Eventos y Comportamientos Dinámicos

Uso de Bibliotecas y Frameworks (si aplicable)

Consumo de Datos desde el Backend

Configuración de Conexiones al Backend

Obtención y Presentación de Datos

Actualización en Tiempo Real (si aplicable)

Interacción Usuario-Interfaz

Manejo de Formularios y Validación de Datos

Implementación de Funcionalidades Interactivas

Mejoras en la Experiencia del Usuario

Pruebas y Depuración del Frontend

Diseño de Casos de Prueba de Frontend

Pruebas de Usabilidad

Depuración de Errores y Optimización del Código

Implementación de la Lógica de Negocio en el Frontend

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

Validación de Datos y Reglas de Negocio en el Frontend

Integración con el Backend

Verificación de la Comunicación Efectiva con el Backend

Pruebas de Integración Frontend-Backend

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.