PRIMERA ENTREGA PROYECTO

ANALIZANDO EL UNIVERSO DEL DISCURSO

Realizado por (Grupo 3):

Nicolas Peña Irurita

Juan Pablo Gómez Veira (2231715)

Isaac Piedrahita (2226259)

William Botero (2225155)

DOCENTE: JHON EDER MASSO DAZA

ALMACENAMIENTO DE DATOS

PROGRAMA DE INGENIERÍA DE DATOS E INTELIGENCIA ARTIFICIAL

FACULTAD DE INGENIERÍA

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE

SANTIAGO DE CALI

2024

Analizando el universo del discurso

Introducción

El sistema DocTIC es una plataforma de almacenamiento y gestión de documentos técnicos que facilita a los usuarios la publicación y revisión de contenido en áreas especializadas como desarrollo web, bases de datos, y programación. Además, proporciona herramientas para la interacción a través de valoraciones y comentarios, y gestiona la seguridad del acceso mediante la autenticación de usuarios y la recuperación de contraseñas.

Propósito del sistema

El objetivo de DocTIC es ofrecer un espacio centralizado en este caso una base de datos para la publicación de documentos técnicos y fomentar la colaboración a través de comentarios y valoraciones. El sistema también permite a los usuarios gestionar sus credenciales y recuperar contraseñas olvidadas, asegurando un acceso seguro a la plataforma.

Usuarios

Los usuarios del sistema incluyen profesionales técnicos y académicos que buscan compartir y consultar documentos especializados. Los administradores tienen roles adicionales en la gestión del sistema y monitoreo de la actividad.

Contexto técnico

El sistema utilizará MySQL Workbench para la gestión de bases de datos y ofrecerá una interfaz de usuario accesible para la interacción con el contenido. Se implementarán medidas de seguridad como el cifrado de contraseñas.

Problemas y soluciones propuestas

Estimamos que los desafíos principales incluyen la seguridad en la gestión de contraseñas y el rendimiento con grandes volúmenes de documentos. Estas cuestiones se abordarán mediante técnicas de cifrado robustas y optimización de consultas de base de datos.

Creación del Modelo Entidad-Relación (MER)

El Modelo Entidad-Relación (MER) es una herramienta fundamental en el diseño de bases de datos que permite representar gráficamente las entidades involucradas en un sistema y las relaciones entre ellas. Para el proyecto DocTIC, se ha diseñado un MER que refleja las necesidades de almacenamiento y gestión de datos del sistema.

Las entidades definidas en el modelo incluyen:

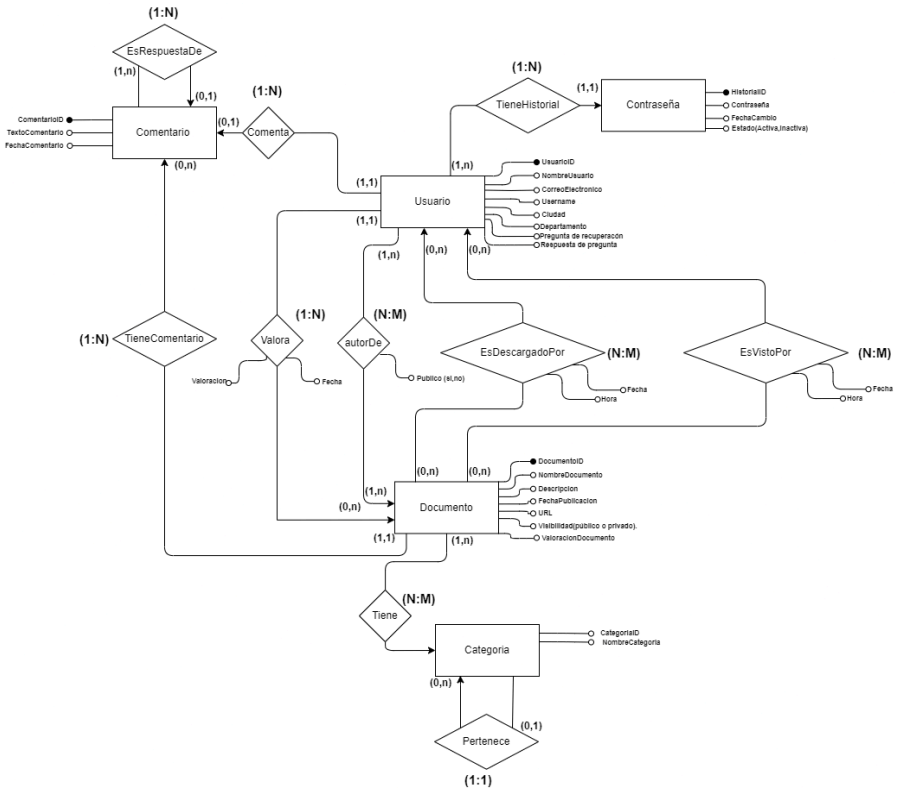
- **Usuario**: Representa a los usuarios del sistema, estos pueden ser autores de documentos, valoradores o comentaristas. Se incluyen atributos como nombre de usuario, correo electrónico, ciudad, departamento, pregunta secreta y respuesta secreta.
- **Documento**: Contiene información sobre los documentos publicados en la plataforma. Los atributos incluyen nombre del documento, descripción, fecha de publicación, URL, visibilidad y valoración.
- **Comentario**: Permite a los usuarios agregar comentarios a los documentos y responder a otros comentarios. Incluye atributos como fecha, texto del comentario, y un campo que indica si es un comentario en respuesta a otro.
- **Categoría**: Define las categorías de los documentos (ej. desarrollo web, bases de datos, etc.), con una relación jerárquica en donde una categoría puede pertenecer a otra.
- **Historial de contraseñas**: Gestiona el historial de contraseñas de los usuarios, asegurando que no se utilicen contraseñas anteriores.

Objetivo del MER

El objetivo principal del MER es organizar los datos de manera que se minimice la redundancia y se mantenga la integridad de la base de datos, permitiendo así una gestión eficiente de los usuarios, documentos y sus interacciones.

Presentación del MER

https://drive.google.com/file/d/1tiLnX5vJh1YrgSqSwqttAxOS-rDnyYRT/view



Usuario:

- **Descripción**: Representa a los usuarios registrados en el sistema. Cada usuario tiene atributos como UsuarioID, NombreUsuario, CorreoElectronico, Username, Ciudad, Departamento, PreguntaRecuperacion, y RespuestaRecuperacion.

Relaciones:

- Tiene un historial de contraseñas (TieneHistorial) que está representado en la entidad **Contraseña**.
- Puede comentar en documentos (Comenta) y valorar documentos (Valora).
- Puede ser autor de uno o más documentos (autorDe).
- Los documentos pueden ser descargados (EsDescargadoPor) y vistos (EsVistoPor) por usuarios.

Contraseña:

- **Descripción:** Almacena el historial de contraseñas de cada usuario, permitiendo registrar cambios y gestionar la seguridad de las cuentas. Atributos como HistorialID, Contraseña, FechaCambio, y Estado (activa/inactiva) están incluidos.

- **Relaciones:**
    - Cada contraseña pertenece a un solo usuario (TieneHistorial).

Documento:

- **Descripción:** Representa los documentos disponibles en el sistema. Cada documento tiene atributos como DocumentoID, NombreDocumento, Descripcion, FechaPublicacion, URL, Visibilidad, y ValoracionDocumento.
- **Relaciones:**
    - Un documento puede pertenecer a una o más categorías (Tiene), y cada documento tiene una clave foránea hacia la entidad **Categoría**.
    - Los documentos pueden ser valorados (Valora), comentados (TieneComentario), descargados (EsDescargadoPor) y vistos (EsVistoPor) por los usuarios.

Comentario:

- **Descripción:** Contiene los comentarios realizados por los usuarios en los documentos. Atributos como ComentarioID, TextoComentario, y FechaComentario están presentes.
- **Relaciones:**
    - Un comentario puede ser una respuesta a otro comentario (EsRespuestaDe), lo que permite estructurar hilos de discusión.
    - Los comentarios se relacionan con un usuario que los realiza (Comenta) y un documento al que pertenecen (TieneComentario).

Categoría:

- **Descripción:** Representa las categorías a las que pueden pertenecer los documentos. Atributos como CategoriaID y NombreCategoria están presentes.
- **Relaciones:**
    - Cada categoría puede pertenecer a una metacategoría (Pertenece), lo que permite una jerarquía de categorías.
    - Una categoría puede estar asociada a múltiples documentos (Tiene).

Valora:

- **Descripción:** Entidad intermedia que captura la valoración de los documentos por parte de los usuarios. Contiene atributos como Valoracion y Fecha.
- **Relaciones:**
    - Relaciona la entidad **Usuario** con la entidad **Documento**.

TieneComentario:

- **Descripción:** Relación que conecta los documentos con sus comentarios. Facilita la consulta de todos los comentarios asociados a un documento.

- **Relaciones:**
    - Relaciona la entidad **Documento** con la entidad **Comentario**.

EsDescargadoPor:

- **Descripción:** Entidad intermedia que registra cada vez que un documento es descargado por un usuario. Almacena la fecha y la hora de la descarga.
- **Relaciones:**
    - Conecta la entidad **Documento** con la entidad **Usuario**.

EsVistoPor:

- **Descripción:** Similar a EsDescargadoPor, pero registra las vistas de documentos. Contiene la fecha y hora en que un documento fue visto por un usuario.
- **Relaciones:**
    - Relaciona la entidad **Documento** con la entidad **Usuario**.

Pertenece:

- **Descripción:** Relación que establece la pertenencia de una categoría a una metacategoría. Esto organiza las categorías de manera jerárquica.
- **Relaciones:**
    - Conecta la entidad **Categoría** con sí misma para formar la jerarquía.

Modelo Relacional (MR)

El Modelo Relacional (MR) es la representación lógica de los datos que se almacenarán en la base de datos DocTIC. El modelo organiza la información en tablas relacionadas entre sí, asegurando que los datos se almacenen de manera eficiente y se puedan recuperar de forma coherente.

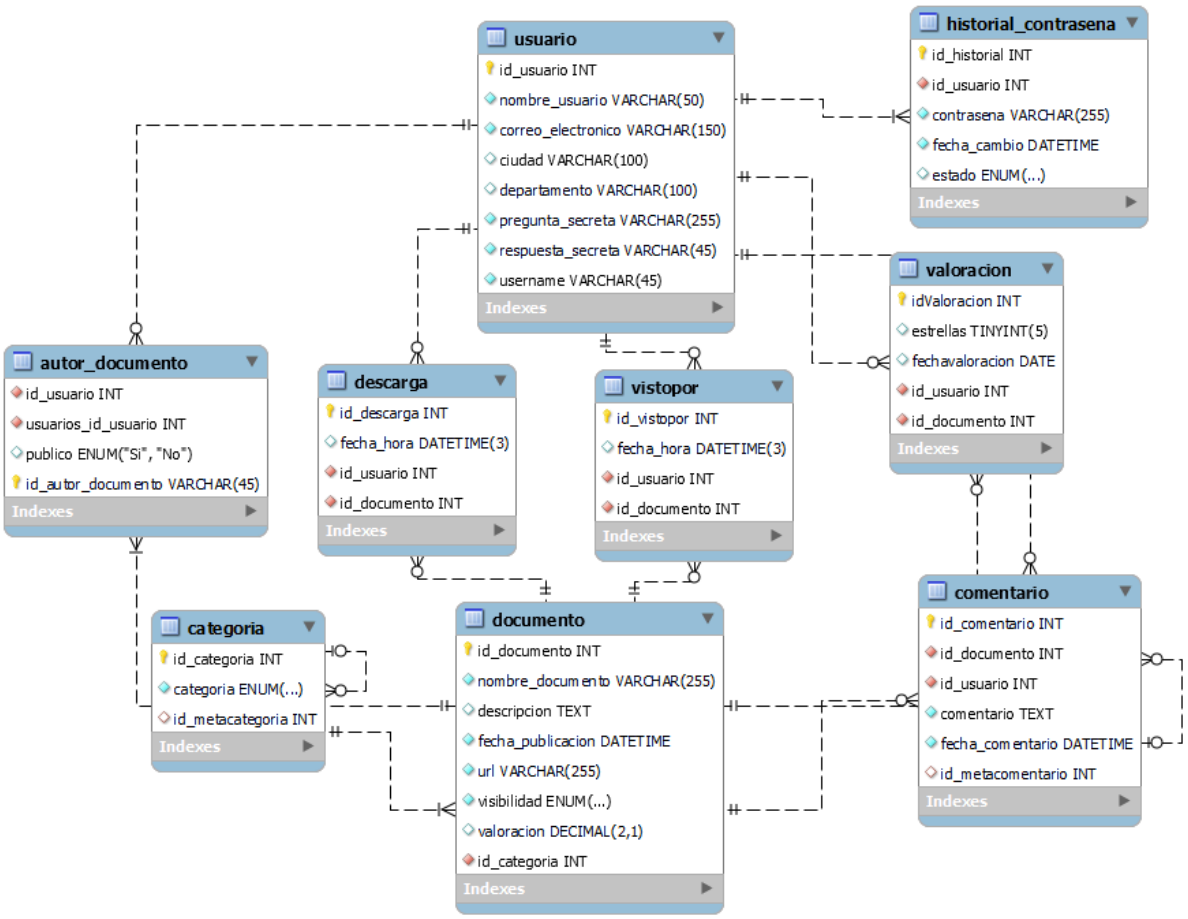
Objetivo del MR

El MR tiene como objetivo estructurar la base de datos DocTIC, definiendo tablas que representan las entidades del sistema y las relaciones entre ellas. Esto facilita la gestión de los datos y asegura que se mantenga la integridad referencial.

Presentación del MR

Derivado del MER, el Modelo Relacional (MR) de DocTIC se estructuró para facilitar la implementación física de la base de datos en MySQL. Cada entidad del MER se transformó en una tabla relacional, con las siguientes claves principales y foráneas:

- **Usuario (usuario)**: Tabla que almacena la información personal y de autenticación de los usuarios. Incluye índices únicos en nombre_usuario y correo_electronico para garantizar la unicidad y mejorar el rendimiento de las consultas.
- **Documento (documento)**: Tabla que almacena los documentos con atributos como id_documento, nombre_documento, fecha_publicacion, url, y visibilidad. Se incluye una clave foránea que referencia la categoría (id_metacategoria).
- **Comentario (comentario)**: Tabla que almacena los comentarios de los usuarios sobre los documentos. Incluye referencias a id_documento y id_usuario, asegurando que cada comentario esté relacionado con un documento y un usuario específico.
- **Autor (autor_documento)**: Como un tiene que ser un usuario que lo sube el autor del documento, puede pasar que 2 o más usuarios sean también autores, para eso se tiene el atributo si ese autor fue o no el que lo público. Además tiene una primer key para establecer la relacion entre el autor y el documento.
- **Valoración (valoracion)**: Tabla que registra las valoraciones de los documentos por parte de los usuarios, con claves foráneas hacia id_documento y id_usuario.
- **Categoría (categoría)**: Tabla que define las categorías disponibles para los documentos, con una estructura jerárquica que permite la categorización anidada.
- **Historial de contraseña (historial_contrasena)**: Tabla que registra las contraseñas históricas de los usuarios, asegurando la no repetición de contraseñas antiguas.
- **Descarga y Vista (descarga, vistopor)**: Tablas que registran la actividad de los usuarios en cuanto a la descarga y visualización de documentos, permitiendo un seguimiento detallado de la interacción de los usuarios con el contenido.



Script SQL

El SQL Script fue generado utilizando la funcionalidad de "Forward Engineering" de MySQL Workbench, para crear todas las tablas y establecer las relaciones entre ellas en la base de datos doctic

Puede observar el script más a fondo Aquí

Este script es esencial para la implementación física de la base de datos, pues nos permite la creación automatizada de las tablas y la configuración de todas las restricciones y relaciones necesarias a partir del **MR**.

Consultas SQL

Se diseñaron unas consultas SQL para extraer y analizar datos clave de la base de datos del proyecto DocTIC. Las consultas se enfocan en aspectos fundamentales como la gestión de documentos, la participación de los usuarios y las valoraciones. Cada consulta ha sido explicada brevemente para facilitar su comprensión y aplicación línea por línea.

Cantidad de Documentos por Categoría

```
SELECT
    (SELECT c.categoria FROM categoria c WHERE c.id_categoria =
d.id_categoria) AS categoria,
    COUNT(d.id_documento) AS cantidad_documentos
FROM
    documento d
GROUP BY
    d.id_categoria
ORDER BY
    cantidad_documentos DESC;
```

Explicación:

- **SELECT:** La subconsulta extrae el nombre de la categoría correspondiente al `id_categoria` de la tabla `documento`. Esto se hace para cada documento.
- **COUNT(d.id_documento):** Cuenta cuántos documentos hay en cada categoría.
- **GROUP BY d.id_categoria:** Agrupa los documentos por categoría, lo que permite contar cuántos documentos pertenecen a cada una.
- **ORDER BY cantidad_documentos DESC:** Ordena los resultados en orden descendente según la cantidad de documentos, de modo que las categorías con más documentos aparezcan primero.

Resultado: Esta consulta te da un resumen de cuántos documentos hay en cada categoría, ordenados de mayor a menor cantidad.

Usuarios que han visto más Documentos de una Categoría Específica (Seguridad Informática)

```
SELECT
    (SELECT u.nombre_usuario FROM usuario u WHERE u.id_usuario =
v.id_usuario) AS nombre_usuario,
    (SELECT c.categoria FROM categoria c WHERE c.id_categoria =
```

```
d.id_categoria) AS categoria,
    COUNT(v.id_vistopor) AS total_vistos
FROM
    vistopor v,
    documento d
WHERE
    v.id_documento = d.id_documento
    AND d.id_categoria = (SELECT id_categoria FROM categoria WHERE
categoria = 'seguridad informática')
GROUP BY
    v.id_usuario, d.id_categoria
ORDER BY
    total_vistos DESC;
```

Explicación:

- SELECT: La primera subconsulta obtiene el nombre del usuario, y la segunda subconsulta obtiene el nombre de la categoría del documento.
- COUNT(v.id_vistopor): Cuenta cuántas veces cada usuario ha visto documentos en la categoría "seguridad informática".
- WHERE v.id_documento = d.id_documento: Filtra para asegurar que estamos considerando las visualizaciones correctas.
- AND d.id_categoria = (subconsulta): Filtra para incluir solo los documentos en la categoría "seguridad informática".
- GROUP BY v.id_usuario, d.id_categoria: Agrupa los resultados por usuario y categoría.
- ORDER BY total_vistos DESC: Ordena los usuarios por la cantidad de documentos vistos, de mayor a menor.

Resultado: Esta consulta te muestra qué usuarios han visto más documentos en la categoría "seguridad informática", permitiendo identificar a los usuarios más interesados en ese tema.

Documentos Nunca Visualizados por Usuarios

```
SELECT
    nombre_documento
FROM
    documento
WHERE
    id_documento NOT IN (SELECT id_documento FROM vistopor);
```

Explicación:

- WHERE id_documento NOT IN: Filtra para obtener los documentos cuyo `id_documento` no aparece en la tabla `vistopor`, es decir, documentos que nunca han sido visualizados por ningún usuario.

Resultado: Lista los documentos que no han sido visualizados por ningún usuario, lo que es útil para identificar contenido que podría necesitar más promoción.

Promedio de Valoración por Documento

```
SELECT
    (SELECT d.nombre_documento FROM documento d WHERE
d.id_documento = v.id_documento) AS nombre_documento,
    AVG(v.estrellas) AS promedio_valoracion
FROM
    valoracion v
GROUP BY
    v.id_documento;
```

Explicación:

- SELECT: La subconsulta obtiene el nombre del documento correspondiente a cada valoración.

- AVG(v.estrellas): Calcula el promedio de las estrellas que ha recibido cada documento.

- GROUP BY v.id_documento: Agrupa los resultados por documento.

Resultado: Muestra el promedio de las valoraciones para cada documento, ayudando a identificar cuáles son los documentos mejor valorados.

Documentos Publicados en el Último Mes con sus Valoraciones

```
SELECT
    nombre_documento,
    fecha_publicacion,
    valoracion
FROM
    documento
WHERE
    fecha_publicacion >= NOW() - INTERVAL 1 MONTH
ORDER BY
    fecha_publicacion DESC;
```

Explicación:

- WHERE fecha_publicacion >= NOW() - INTERVAL 1 MONTH: Filtra los documentos publicados en el último mes.

- ORDER BY fecha_publicacion DESC: Ordena los documentos por la fecha de publicación, mostrando primero los más recientes.

Resultado: Lista los documentos publicados en el último mes, junto con sus valoraciones, ordenados por fecha de publicación.

Usuarios con el Mayor Número de Descargas

```
SELECT
    (SELECT u.nombre_usuario FROM usuario u WHERE u.id_usuario =
d.id_usuario) AS nombre_usuario,
    COUNT(d.id_descarga) AS total_descargas
FROM
    descarga d
GROUP BY
    d.id_usuario
ORDER BY
    total_descargas DESC;
```

Explicación:

- **SELECT:** La subconsulta obtiene el nombre del usuario correspondiente a cada descarga.
- **COUNT(d.id_descarga):** Cuenta el número de descargas realizadas por cada usuario.
- **GROUP BY d.id_usuario:** Agrupa las descargas por usuario.
- **ORDER BY total_descargas DESC:** Ordena los resultados para mostrar primero los usuarios con más descargas.

Resultado: Identifica a los usuarios más activos en términos de descargas, permitiendo enfocar esfuerzos en retener a estos usuarios.

Documentos con la Mayor Cantidad de Comentarios

```
SELECT
    (SELECT nombre_documento FROM documento d WHERE d.id_documento
= c.id_documento) AS nombre_documento,
    COUNT(c.id_comentario) AS total_comentarios
FROM
    comentario c
GROUP BY
    c.id_documento
ORDER BY
    total_comentarios DESC;
```

Explicación:

- **SELECT:** La subconsulta obtiene el nombre del documento para cada comentario.
- **COUNT(c.id_comentario):** Cuenta el número de comentarios realizados en cada documento.
- **GROUP BY c.id_documento:** Agrupa los comentarios por documento.
- **ORDER BY total_comentarios DESC:** Ordena los documentos por el número de comentarios, de mayor a menor.

Resultado: Muestra qué documentos han generado más discusión, lo cual es útil para identificar contenido popular o controvertido.

Usuarios que han dado Valoraciones de 5 Estrellas

```
SELECT
    (SELECT u.nombre_usuario FROM usuario u WHERE u.id_usuario =
v.id_usuario) AS nombre_usuario,
```

```
    COUNT(*) AS valoraciones_cinco_estrellas
FROM
    valoracion v
WHERE
    estrellas = 5
GROUP BY
    v.id_usuario
HAVING
    valoraciones_cinco_estrellas > 0
ORDER BY
    valoraciones_cinco_estrellas DESC;
```

Explicación:

- **SELECT:** La subconsulta obtiene el nombre del usuario que ha dado valoraciones.
- **WHERE estrellas = 5:** Filtra para incluir solo las valoraciones de 5 estrellas.
- **GROUP BY v.id_usuario:** Agrupa las valoraciones por usuario.
- **HAVING valoraciones_cinco_estrellas > 0:** Muestra solo los usuarios que han dado al menos una valoración de 5 estrellas.
- **ORDER BY valoraciones_cinco_estrellas DESC:** Ordena por la cantidad de valoraciones de 5 estrellas, de mayor a menor.

Resultado: Identifica a los usuarios que más frecuentemente otorgan valoraciones máximas, lo cual puede ser útil para detectar tendencias o preferencias.

Documentos que No Han Recibido Comentarios

```
SELECT
    nombre_documento
FROM
    documento
WHERE
    id_documento NOT IN (SELECT id_documento FROM comentario);
```

Explicación:

- **WHERE id_documento NOT IN:** Filtra para encontrar documentos que no tienen ningún comentario asociado.

Resultado: Lista los documentos que no han recibido ningún comentario, lo que puede indicar que no han generado interés o no han sido suficientemente visibles.

Usuarios que han Visualizado y Luego Descargado Documentos

```
SELECT
    (SELECT nombre_usuario FROM usuario u WHERE u.id_usuario =
v.id_usuario) AS nombre_usuario
FROM
    vistopor v
WHERE
    id_documento IN (SELECT id_documento FROM descarga WHERE
v.id_usuario = descarga.id_usuario);
```

Explicación:

- **WHERE id_documento IN:** Filtra para encontrar usuarios que han visto un documento y luego lo han descargado.
- **SELECT:** La subconsulta obtiene el nombre del usuario.

Resultado: Identifica usuarios que no solo han visualizado documentos, sino que también han descargado esos documentos, lo cual puede ser un indicador de alto interés en el contenido.