

SUCCESS MINDSET

MODELO DE NEGOCIO

BASE DE DATOS

PROYECTO FINAL

Alumno: Felipe Lopez Rios
Curso SQL Coderhouse
Comision 47370



ÍNDICE

2	– Descripción de la temática
10	– Desarrollo de la database
11	– Listado de tablas
24	– Listado de vistas
32	– Listado de funciones
39	– Listado de triggers y tablas de auditoría
49	– Reverse engineer
50	– Scripts
51	– Tecnologías utilizadas

DESCRIPCIÓN DE LA TEMÁTICA

INTRODUCCIÓN

En un mundo cada vez más enfocado en el crecimiento personal y el desarrollo de habilidades, el acceso a recursos valiosos se volvió fundamental. Es por esto que propongo "Success Mindset".

"Success Mindset" surge como respuesta a la creciente demanda de una plataforma que permita a los lectores entusiastas acceder a libros que fomenten el crecimiento y la superación personal.

Este proyecto de base de datos tiene como objetivo crear una infraestructura sólida para un e-commerce de libros dedicado a este nicho, brindando a los usuarios la oportunidad de comprar y vender libros nuevos o usados que contribuyan al desarrollo de una mentalidad de éxito.

OBJETIVO

El objetivo principal de este proyecto es proporcionar un espacio digital accesible, seguro y eficiente donde los usuarios puedan explorar, adquirir y vender libros que contribuyan a su desarrollo personal y el logro de sus metas. Con esta base de datos que estoy construyendo, busco simplificar y mejorar la experiencia de búsqueda y compra de libros que fomenten una mentalidad de éxito.

Objetivos específicos:

- Gestión de inventario eficiente
- Mejorar la experiencia de usuario
- Fomentar una comunidad
- Generar informes de ventas y estadísticas para el negocio

SITUACIÓN PROBLEMÁTICA

En un mundo donde el acceso a información y recursos es prácticamente indispensable para el crecimiento personal y el desarrollo de habilidades, la necesidad de una plataforma que aborde las carencias existentes en la adquisición y venta de libros de desarrollo personal de una manera mucho más práctica y sencilla es evidente.

En los siguientes ítems describiré las problemáticas y brechas que una sólida base de datos en este proyecto busca abordar.

- Necesidad de facilitar la gestión y un amplio acceso a libros de desarrollo profesional: la situación problemática a la que nos enfrentamos es la falta de una plataforma integral y eficiente que facilite tanto a vendedores como a compradores el proceso de gestión y acceso a libros de desarrollo profesional.

Si bien existen numerosos recursos impresos y digitales disponibles en este nicho, así como diversas tiendas de e-commerce que permite la compra y venta de estos artículos, la dispersión de la información, la falta de opciones de compra y venta confiables y la ausencia de una comunidad dedicada son problemáticas significativas que este proyecto busca abordar.

- Escasa información para la toma de decisiones: tanto vendedores como administradores carecen de información valiosa sobre el rendimiento del negocio y las preferencias de los compradores, dificultando la toma de decisiones estratégicas. La implementación

de una base de datos sólida y bien diseñada es esencial para abordar estas problemáticas.

- Dificultad en la búsqueda de libros relevantes: los compradores entusiastas, así como las personas que se encuentran en proceso de introducción en el mundo del desarrollo personal, enfrentan obstáculos a la hora de encontrar libros específicos. La falta de una plataforma centralizada hace que la búsqueda sea engorrosa y poco eficiente.

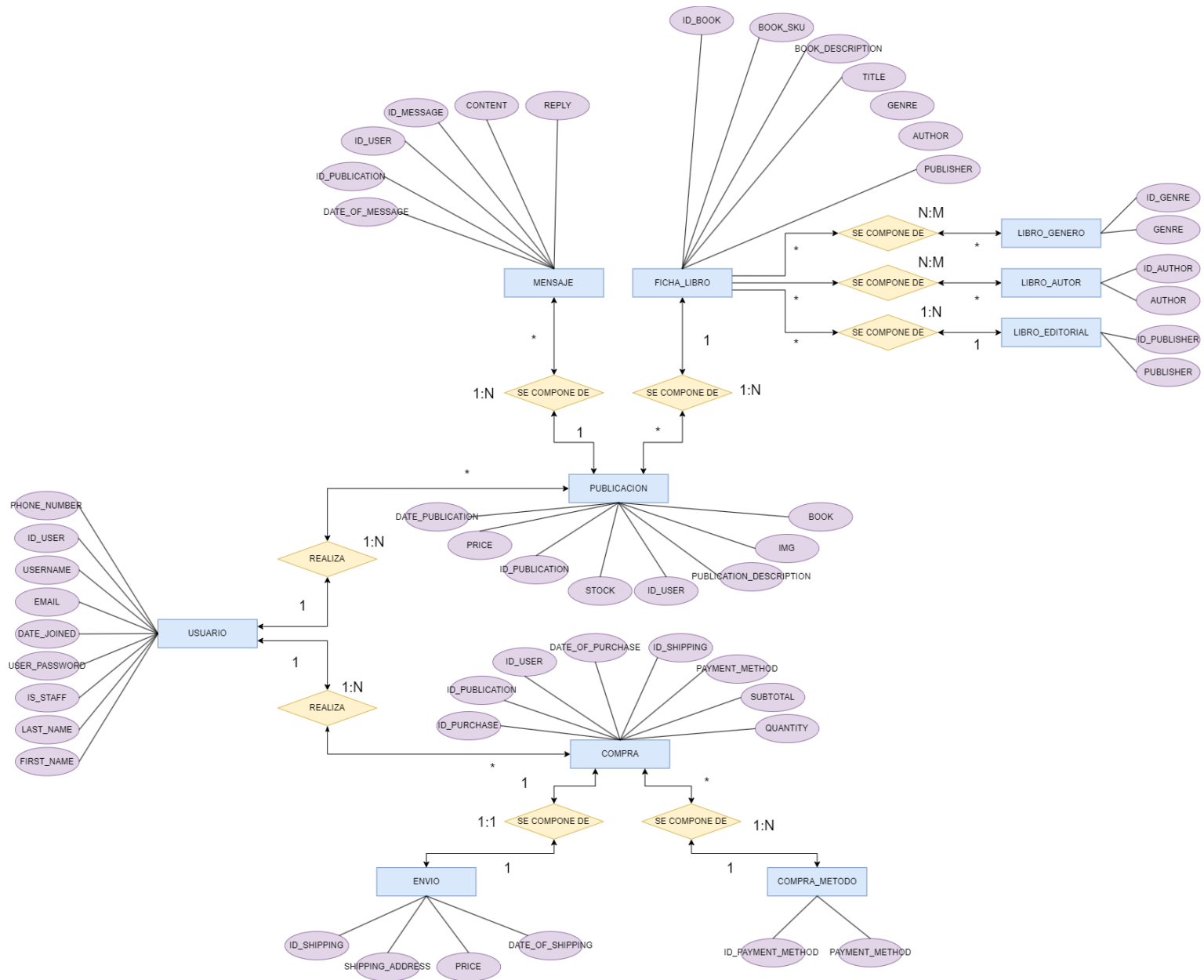
MODELO DE NEGOCIO

La base de datos se encuentra en desarrollo para ser utilizada por "Success Mindset", sustentándose en un sólido modelo de negocio diseñado para satisfacer las necesidades de vendedores como de compradores, al tiempo que genera ingresos sostenibles.

El modelo de negocio que planteo conlleva los siguientes componentes clave:

- Venta directa de libros: es la columna vertebral del negocio. Cada transacción genera ingresos para el mismo mediante comisiones por venta.
- Publicidad y promoción: se ofrecen opciones de publicidad y promoción pagadas a los vendedores que desean destacar sus libros en la plataforma, permitiéndole a los mismos aumentar la visibilidad de sus productos.
- Generación de informes y estadísticas: la plataforma generará informes detallados sobre las ventas, preferencias de los compradores y tendencias en el desarrollo personal/profesional. Estos informes podrán ser vendidos a editores, autor y otros actores del mercado que deseen acceder a datos de mercado que puedan tener valor significativo para los mismos.

DIAGRAMA ENTIDAD-RELACIÓN (DER)



Para acceder al DER en una mayor escala: [en el siguiente link](#)

DESARROLLO DE LA DATABASE

LISTADO DE TABLAS

USUARIO								
id_user INT PK	username VARCHAR(30) UNIQUE	phone_number VARCHAR(15)	email VARCHAR(60)	date_joined DATETIME	password VARCHAR(255)	first_name VARCHAR(30)	last_name VARCHAR(60)	is_staff BOOL

Descripción: esta tabla representa a la entidad "Usuario" del E-commerce, el cual posee los datos de toda persona que se registre y utilice este servicio.

Listado de campos:

- id_user INT PK AUTO_INCREMENT NOT NULL: este campo representa el "ID del usuario".
- username VARCHAR(30) UNIQUE NOT NULL: este campo representa el "nombre de usuario".
- phone_number VARCHAR(15) : este campo representa el "número de teléfono de contacto" del usuario.
- email VARCHAR(60) NOT NULL : este campo representa la "dirección de correo electrónico de contacto" del usuario.
- date_joined DATETIME DEFAULT CURRENT_TIMESTAMP: este campo representa la "fecha de creación de la cuenta" del usuario.
- password VARCHAR(255) NOT NULL : este campo representa la "contraseña" de la cuenta del usuario.
- first_name VARCHAR(30) NOT NULL : este campo representa el "nombre" del usuario.

- last_name VARCHAR(60) NOT NULL : este campo representa el/los "apellido/s" del usuario
- is_staff BOOL NOT NULL DEFAULT 0: este campo representa si el usuario "es staff". Se utiliza para crear superusuarios que tengan acceso al panel administrativo y otros permisos especiales del e-commerce con el fin de que los desarrolladores puedan hacer pruebas en la pagina

Script de creación:

```
CREATE Table usuario (
  id_user INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  username VARCHAR(30) UNIQUE NOT NULL,
  phone_number VARCHAR(15),
  email VARCHAR(60) NOT NULL,
  date_joined DATETIME DEFAULT CURRENT_TIMESTAMP
,
  user_password VARCHAR(255) NOT NULL,
  first_name VARCHAR(30) NOT NULL,
  last_name VARCHAR(60) NOT NULL,
  is_staff BOOL DEFAULT 0 NOT NULL
);
```

PUBLICACION							
id_publication INT PK	id_user INT FK	price FLOAT	stock INT	description TEXT	date_publication DATETIME	img VARCHAR(255)	book INT FK

Descripción: esta tabla representa a la entidad "Publicación" del E-commerce, la cual es realizada por los usuarios para realizar sus ventas en la página.

Listado de campos:

- id_publication INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id de la publicación"
- id_user INT FK NOT NULL : este campo representa la relacion "id del usuario" con la tabla USUARIO
- price FLOAT NOT NULL : este campo representa el "precio" de la publicación que el vendedor le establece
- stock INT NOT NULL : este campo representa la cantidad de "stock" de libros de la publicación
- description TEXT : este campo representa la "descripción de la publicación"
- date_publication DATETIME DEFAULT CURRENT_TIMESTAMP : este campo representa "la fecha de publicación"
- book INT FK NOT NULL : este campo representa la relacion "ficha del libro" con la tabla FICHA_LIBRO
- img VARCHAR(255) DEFAULT (URL de imagen por defecto) : este campo representa "la URL donde va a estar alojada la imagen de la publicación"

Script de creación:

```
CREATE Table publicacion (
  id_publicacion INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  id_user INT NOT NULL,
  book INT NOT NULL,
  price FLOAT NOT NULL,
  stock INT NOT NULL,
  publication_description TEXT,
  date_publication DATETIME DEFAULT CURRENT_TIMESTAMP,
  img VARCHAR(255) DEFAULT 'https://i.imgur.com/tOj1c0U.jpg',
  CONSTRAINT fk_user FOREIGN KEY(id_user) REFERENCES usuario(id_user) ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_book FOREIGN KEY(book) REFERENCES ficha_libro(id_book) ON UPDATE CASCADE ON DELETE CASCADE
);
```

FICHA_LIBRO				
id_book INT PK	publisher INT FK	sku INT	description TEXT	title VARCHAR(80)

Descripción: esta tabla representa a la entidad "Ficha_libro" (es decir, las características del libro de la publicación a la que está relacionada). Esta entidad posee las características del libro, tal así como su descripción, su título, su código ISBN (sku), etcétera.

Listado de campos:

- id_book INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id del libro".
- publisher INT FK NOT NULL : este campo representa la relación "editorial del libro" con la tabla LIBRO_EDITORIAL
- sku INT : este campo representa el "código SKU" del libro.
- description TEXT : este campo representa "la descripción del libro".
-

- title VARCHAR(80) NOT NULL : este campo representa el "título del libro".

Script de creación:

```
CREATE Table ficha_libro (  
  id_book INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
  publisher INT NOT NULL,  
  sku INT, -- Numero ISBN del libro  
  book_description TEXT,  
  title VARCHAR(80) NOT NULL,  
  CONSTRAINT fk_publisher FOREIGN KEY(publisher) REFERENCES  
  libro_editorial(id_publisher) ON UPDATE CASCADE ON DELETE CASCADE  
);
```

LIBRO_GENERO	
id_genre INT PK	genre VARCHAR(30)

Descripción: esta tabla se utiliza para relacionar la entidad "Ficha_libro" con su género, y evitar así la redundancia de los datos

Listado de campos:

- id_genre INT PK AUTO_INCREMENT NOT NULL: este campo representa el "id del género".
- genre VARCHAR(30) UNIQUE NOT NULL : este campo representa "el género" del libro

Script de creación:

```
CREATE Table libro_genero (  
    id_genre INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    genre VARCHAR(30) UNIQUE NOT NULL  
);
```

LIBRO_AUTOR	
id_author INT PK	author VARCHAR(60)

Descripción: esta tabla se utiliza para relacionar la entidad "Ficha_libro" con su autor, y evitar así la redundancia de los datos

Listado de campos:

- id_author INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id del autor".
- author VARCHAR(60) UNIQUE NOT NULL : este campo representa "el nombre del autor" que escribe el libro

Script de creación:

```
CREATE Table libro_autor (  
    id_author INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    author VARCHAR(60) UNIQUE NOT NULL  
);
```

LIBRO_EDITORIAL	
id_publisher INT PK	publisher VARCHAR(30)

Descripción: esta tabla se utiliza para relacionar la entidad "Ficha_libro" con la editorial, y evitar así la redundancia de los datos

Listado de campos:

- id_publisher INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id de la editorial".
- publisher VARCHAR(30) UNIQUE NOT NULL : este campo representa "el nombre de la editorial" que publica el libro

Script de creación:

```
CREATE Table libro_editorial (  
    id_publisher INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    publisher VARCHAR(30) UNIQUE NOT NULL  
);
```

LIBRO_GENERO_RELACION		
id_relacion INT PK	id_libro INT FK NOT NULL	id_genero INT FK NOT NULL

Descripción: esta tabla se utiliza como intermediaria para expresar la relación muchos a muchos entre la tabla “ficha_libro” y “libro_genero”

Listado de campos:

- id_relacion INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id de la relacion".
- id_libro INT FK NOT NULL: este campo representa el "id del libro".
- Id_genero INT FK NOT NULL: este campo representa el "id del genero" (proviene de "libro_genero")

Script de creación:

```
CREATE Table libro_genero_relacion (  
    id_relacion INT PRIMARY KEY AUTO_INCREMENT,  
    id_libro INT NOT NULL,  
    id_genero INT NOT NULL,  
    CONSTRAINT fk_libro_genero FOREIGN KEY (id_libro) REFERENCES  
ficha_libro(id_book) ON UPDATE CASCADE ON DELETE CASCADE,  
    CONSTRAINT fk_genero FOREIGN KEY (id_genero) REFERENCES  
libro_genero(id_genre) ON UPDATE CASCADE ON DELETE CASCADE  
);
```

LIBRO_AUTOR_RELACION		
id_relacion INT PK	id_libro INT FK NOT NULL	id_autor INT FK NOT NULL

Descripción: esta tabla se utiliza como intermediaria para expresar la relación muchos a muchos entre la tabla “ficha_libro” y “libro_autor”

Listado de campos:

- id_relacion INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id de la relacion".
- id_libro INT FK NOT NULL: este campo representa el "id del libro".
- id_autor INT FK NOT NULL: este campo representa el "id del autor" (proviene de "libro_autor")

Script de creación:

```
CREATE Table libro_autor_relacion (  
    id_relacion INT PRIMARY KEY AUTO_INCREMENT,  
    id_libro INT NOT NULL,  
    id_autor INT NOT NULL,  
    CONSTRAINT fk_libro_autor FOREIGN KEY (id_libro) REFERENCES  
ficha_libro(id_book) ON UPDATE CASCADE ON DELETE CASCADE,  
    CONSTRAINT fk_autor FOREIGN KEY (id_autor) REFERENCES  
libro_autor(id_author) ON UPDATE CASCADE ON DELETE CASCADE  
);
```

MENSAJE					
id_message INT PK	id_user INT FK	id_publication INT FK	content TEXT	reply TEXT	date_of_message DATETIME

Descripción: esta tabla representa a la entidad "Mensaje". Esta entidad representa a los mensajes que un usuario puede realizar en una publicación (a la cual el mismo está relacionado). Esta entidad posee características propias de un mensaje, tal así como su contenido, la respuesta que el dueño de la publicación realiza sobre el mismo, y su fecha de publicación.

Listado de campos:

- id_message INT PK AUTO_INCREMENT NOT NULL: este campo representa el "id del mensaje de la publicación".
- id_user INT FK NOT NULL : este campo representa la relacion "usuario que realizo el comentario" con la tabla USUARIO

- `id_publication INT FK NOT NULL` : este campo representa la relacion "publicación donde se realizó el mensaje" con la tabla PUBLICACION
- `content TEXT NOT NULL` : este campo representa el "contenido del mensaje".
- `reply TEXT` : este campo representa "la respuesta que el dueño de la publicación le realiza al mensaje".
- `date_of_message DATETIME DEFAULT CURRENT_TIMESTAMP` : este campo representa "la fecha de publicación del mensaje"

Script de creación:

```
CREATE Table mensaje (
  id_message INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
  id_user INT NOT NULL,
  id_publication INT NOT NULL,
  content TEXT NOT NULL,
  reply TEXT,
  date_of_message DATETIME DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_sender_user FOREIGN KEY(id_user) REFERENCES usuario(id_user) ON UPDATE CASCADE
ON DELETE CASCADE,
  CONSTRAINT fk_publication FOREIGN KEY(id_publication) REFERENCES publicacion(id_publication)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

COMPRA							
id_purchases INT PK	id_publication INT FK	id_user INT FK	id_shipping INT FK	date_of_purchase DATETIME	payment_method INT FK	quantity INT	subtotal FLOAT

Descripción: esta tabla representa a la entidad "Compra", la cual representa a las compras de los libros (las publicaciones a las cuales esta relacionadas) que realizan los usuarios en el e-commerce. Esta entidad posee características como la fecha en la que se realizó la compra, el método de pago, las claves foráneas que las relacionan con la publicación, entre otras

Listado de campos:

- id_purchase INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id de la compra".
- id_publication INT FK NOT NULL : este campo representa la relacion "publicacion que se ha comprado (producto)" con la tabla PUBLICACION
- id_user INT FK NOT NULL : este campo representa la relacion "usuario que realizo la compra" con la tabla USUARIO
- id_shipping INT FK NOT NULL : este campo representa la relacion "envío correspondiente a la compra" con la tabla ENVIO.
- date_of_purchase DATETIME DEFAULT CURRENT_TIMESTAMP : este campo representa "la fecha de compra de la publicación (producto)"
- payment_method INT FK NOT NULL : este campo representa la relacion "método de pago" con la tabla COMPRA_METODO
- quantity INT NOT NULL : este campo representa la "cantidad comprada de unidad de producto (publicación)."
- subtotal FLOAT NOT NULL : este campo representa el "precio subtotal" de la compra.

Script de creación:

```
CREATE Table compra (
    id_purchase INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    id_publication INT NOT NULL,
    id_user INT NOT NULL,
    id_shipping INT NOT NULL,
    date_of_purchase DATETIME DEFAULT CURRENT_TIMESTAMP,
    payment_method INT NOT NULL,
    quantity INT NOT NULL,
    subtotal FLOAT NOT NULL,
    CONSTRAINT fk_purchase_publication FOREIGN KEY(id_publication) REFERENCES
publicacion(id_publication) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT fk_buyer FOREIGN KEY(id_user) REFERENCES usuario(id_user) ON UPDATE
CASCADE ON DELETE CASCADE,
    CONSTRAINT fk_shipping FOREIGN KEY(id_shipping) REFERENCES envio(id_shipping) ON
UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT fk_payment_method FOREIGN KEY(payment_method) REFERENCES
compra_metodo(id_payment_method) ON UPDATE CASCADE ON DELETE CASCADE
);
```

COMPRA_METODO	
id_payment_method INT PK	payment_method VARCHAR(15)

Descripción: esta tabla se utiliza para relacionar la entidad "COMPRA" con su método de pago, y evitar así la redundancia de los datos.

Listado de campos:

- id_payment_method INT PK AUTO_INCREMENT NOT NULL : este campo representa el "id del método de pago".
-

- author VARCHAR(15) UNIQUE NOT NULL : este campo representa "el método de pago".

Script de creación:

```
CREATE Table compra_metodo (  
    id_payment_method INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    payment_method VARCHAR(15) UNIQUE NOT NULL  
);
```

ENVIO			
id_shipping INT PK	price FLOAT	shipping_address VARCHAR(120)	date_of_shipping DATETIME

Descripción: esta tabla representa a la entidad "Envío", la cual representa al envío posterior a la compra de un libro (una publicación a la cual se encuentra relacionada). Esta entidad posee características tales como el valor del envío, la dirección a la cual se envía, la fecha en la que se realizara, entre otras.

Listado de campos:

- id_shipping INT PK AUTO_INCREMENT NOT NULL: este campo representa el "id del envío".
- price FLOAT NOT NULL : este campo representa "el valor del envío".

- shipping_address VARCHAR(120) NOT NULL : este campo representa "la dirección de destino del envío".
- date_of_shipping DATETIME NOT NULL : este campo representa "la fecha en el que el envío será realizado".

Script de creación:

```
CREATE Table envio (  
    id_shipping INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    price FLOAT NOT NULL,  
    shipping_address VARCHAR(120) NOT NULL,  
    date_of_shipping DATETIME NOT NULL  
);
```

LISTADO DE VISTAS

Vista USUARIO_COMPRAS

Descripción: la vista USUARIO_COMPRAS está diseñada para proporcionar una visualización detallada de los usuarios y sus compras individuales.

Objetivo: el propósito principal de esta vista es permitir a los programadores visualizar y comprender fácilmente la actividad de compra de cada usuario en particular. Al mostrar los detalles de las compras de libros junto con la información del usuario, la vista facilita un análisis detallado de las preferencias de compra de los usuarios individuales

Tablas/Datos: esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

- id_user: El ID único del usuario que realizó la compra.
- username: El nombre de usuario asociado al usuario en cuestión.
- first_name: El nombre del usuario.
- last_name: El apellido del usuario.
- id_purchase: El ID único de la compra realizada.
- title: El título del libro.
- date_of_purchase: La fecha en que se realizó la compra.
- quantity: La cantidad de libros comprados.
- subtotal: El subtotal de la compra realizada.

Las tablas son:

- usuario: Contiene información detallada sobre los usuarios registrados.
- compra: Almacena detalles específicos de las compras realizadas por los usuarios, como la fecha de compra, la cantidad y el subtotal.
- publicacion: Proporciona información sobre las publicaciones específicas de libros, incluidos los precios y el stock.
- ficha_libro: Contiene detalles exhaustivos sobre los libros disponibles

Script de creación:

```
CREATE VIEW usuario_compras AS
  SELECT u.id_user, u.username, u.first_name, u.last_name, c.id_purchase,
 f.title, c.date_of_purchase, c.quantity, c.subtotal
  FROM usuario u
 JOIN compra c ON u.id_user = c.id_user
 JOIN publicacion p ON c.id_publication = p.id_publication
 JOIN ficha_libro f ON p.book = f.id_book;
;
```

Vista INFO_LIBRO

Descripción: esta vista está diseñada para proporcionar una visualización detallada de un libro específico disponible en el sistema. Esta vista combina información esencial de múltiples tablas relacionadas a ficha_libro tales como libro_genero_relacion, libro_autor_relacion y

libro_editorial, para ofrecer una visión completa de los detalles del mismo.

Objetivo: el objetivo de esta vista es permitir un acceso y visualización más sencillo a la información clave de los libros disponibles. Al mostrar detalles como el título del libro, el género, el autor y la editorial, la vista facilita a los usuarios obtener una comprensión clara de la información de cada libro en particular.

Tablas/Datos: esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

- id_book: El ID único del libro en la base de datos.
- title: El título del libro.
- sku: El número ISBN del libro.
- book_description: Una descripción detallada del libro.
- genres: El/los género/s al que pertenece el libro.
- authors: El/los autor/es del libro.
- publisher: La editorial que publicó el libro.

Las tablas serían:

- ficha_libro: Contiene información detallada sobre cada libro en la base de datos, como el título, la descripción y el número ISBN del libro.
- libro_genero_relacion (libro_genero): Almacena información sobre los diferentes géneros literarios a los que pertenecen los libros.
- libro_autor_relacion (libro_autor): Proporciona detalles sobre los autores de los libros disponibles en la base de datos.
- libro_editorial: Contiene información sobre las editoriales que han publicado los libros disponibles en el sistema

Script de creación:

```
CREATE VIEW info_libro AS
  SELECT f.id_book, f.title, f.sku, f.book_description, GROUP_CONCAT(g.genre)
AS genres, GROUP_CONCAT(a.author) AS authors, e.publisher
  FROM ficha_libro f
  JOIN libro_editorial e ON f.publisher = e.id_publisher
  LEFT JOIN libro_genero_relacion gr ON f.id_book = gr.id_libro
  LEFT JOIN libro_genero g ON gr.id_genero = g.id_genre
  LEFT JOIN libro_autor_relacion ar ON f.id_book = ar.id_libro
  LEFT JOIN libro_autor a ON ar.id_autor = a.id_author
  GROUP BY f.id_book;
```

Vista PUBLICACION_MENSAJE

Descripción: esta vista está diseñada para proporcionar una visualización integral de todos los mensajes asociados con cada publicación. Esta vista combina datos importantes de múltiples tablas, incluidas publicacion, mensaje y usuario, para ofrecer una visión detallada de los mensajes relacionados con cada publicación en particular.

Objetivo: el objetivo principal de esta vista es permitir a los programadores visualizar y comprender fácilmente todos los mensajes asociados con cada publicación en el sistema. Al mostrar detalles como el ID de la publicación, el ID de usuario, el contenido del mensaje, la fecha del mensaje y cualquier respuesta asociada, la vista facilita un análisis completo de la interacción y la comunicación relacionadas con cada publicación.

Tablas/Datos: esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

- id_publication: El ID único de la publicación asociada con el mensaje.
- id_user: El ID único del usuario que envió el mensaje.
- first_name: El nombre del usuario que envió el mensaje.
- content: El contenido del mensaje enviado.
- date_of_message: La fecha en que se envió el mensaje.
- reply: Cualquier respuesta asociada con el mensaje, si la hay.

Las tablas serían:

- publicacion: Contiene información detallada sobre las publicaciones.
- mensaje: Almacena todos los mensajes asociados con las publicaciones, incluido el contenido del mensaje, las fechas y cualquier respuesta relacionada.
- usuario: Proporciona detalles sobre los usuarios que han enviado mensajes relacionados con las publicaciones, como el nombre y el ID de usuario

Script de creación:

```
CREATE VIEW publicacion_mensaje AS
    SELECT p.id_publication, u.id_user, u.first_name, m.content,
    m.date_of_message, m.reply
    FROM publicacion p
    JOIN mensaje m ON p.id_publication = m.id_publication
    JOIN usuario u ON m.id_user = u.id_user;
```

Vista USUARIO_COMPRAS_TOTALES

Descripción: esta vista proporciona una visión consolidada del total de las compras realizadas por cada usuario. Esta vista combina datos clave de las tablas usuario y compra para mostrar el precio total de todas las compras realizadas por cada usuario en particular.

Objetivo: el objetivo principal de esta vista es permitir a los programadores obtener una visión global del valor total de las compras realizadas por cada usuario en el sistema. Al mostrar el precio total final de todas las compras realizadas para cada usuario, la vista facilita un análisis rápido y eficiente del consumo de cada cliente individual.

Tablas/Datos: esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

- id_user: El ID único del usuario.
- username: El nombre de usuario asociado al usuario en cuestión.

- first_name: El nombre del usuario.
- last_name: El apellido del usuario.
- total_sales: El monto total de todas las compras realizadas por el usuario, calculado como la suma de los subtotales de todas las compras.

Las tablas serían:

- usuario: Contiene información detallada sobre los usuarios registrados.
- compra: Almacena detalles específicos de las compras realizadas por los usuarios, como la fecha de compra, la cantidad y el subtotal.

Script de creación:

```
CREATE VIEW usuario_compras_totales AS
    SELECT u.id_user, u.username, u.first_name, u.last_name,
    SUM(c.subtotal) AS total_sales
    FROM usuario u
    JOIN compra c ON u.id_user = c.id_user
    GROUP BY u.username;
```

Vista USUARIO_CANTIDAD_COMPRAS

Descripción: esta vista permite visualizar la cantidad de compras realizadas por cada usuario. Esta vista combina datos importantes de las

tablas usuario y compra para mostrar la cantidad total de compras realizadas por cada usuario individual.

Objetivo: el objetivo principal de esta vista es permitir a los programadores obtener una comprensión clara de la cantidad de compras realizadas por cada usuario en particular. Al mostrar la cantidad de compras para cada usuario, la vista facilita un análisis rápido y eficiente del comportamiento de compra de cada cliente individual.


Tablas/Datos: esta vista se compone de los siguientes campos extraídos de las tablas subyacentes:

- id_user: El ID único del usuario en el sistema.
- username: El nombre de usuario asociado al usuario en cuestión.
- email: La dirección de correo electrónico del usuario.
- purchase_count: El número total de compras realizadas por el usuario.

Las tablas son:

- usuario: Contiene información detallada sobre los usuarios registrados.
- compra: Almacena detalles específicos de las compras realizadas por los usuarios, como la fecha de compra, la cantidad y el subtotal.

Script de creación:



```
CREATE VIEW usuario_cantidad_compras AS
  SELECT u.id_user, u.username, u.email, COUNT
(c.id_purchase) AS purchase_count
  FROM usuario u
  LEFT JOIN compra c ON u.id_user = c.id_user
  GROUP BY u.id_user;
```

LISTADO DE FUNCIONES

Función STOCK_GENERO

Descripción: esta función se utiliza para calcular la cantidad total de libros en stock de un género específico. Esta función combina datos esenciales de las tablas publicación y ficha_libro para determinar la cantidad total de libros disponibles en stock para un género particular.

Objetivo: el propósito principal de esta función es proporcionar una manera eficiente de calcular el inventario total de libros disponibles para la venta dentro de un género específico. Al realizar este cálculo, la función permite a los usuarios obtener rápidamente una visión clara de la disponibilidad de libros dentro de un género determinado.

Tablas/Datos: esta función se compone de los siguientes elementos clave:

- Parámetro de entrada: id_genero - Un entero que representa el ID del género para el cual se desea calcular el inventario total.
- Variable local: stock_total - Un entero que almacena el valor total de stock para el género especificado.
- Consulta SQL: La función utiliza una consulta SQL que calcula la suma total del stock de libros para un género específico basado en el parámetro de entrada id_genero.

Esta función opera principalmente en las siguientes tablas:

- publicacion: Contiene información detallada sobre las publicaciones de libros en el sistema, incluidos detalles específicos como el stock y los precios de los libros.
- ficha_libro: Almacena datos exhaustivos sobre los libros disponibles en el sistema, como el género al que pertenece cada libro.

Script de creación:

```
DELIMITER $$
CREATE FUNCTION stock_genero(id_genero INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE stock_total INT;
    SELECT SUM(stock) INTO stock_total
    FROM publicacion p
    JOIN ficha_libro f ON p.book = f.id_book
    JOIN libro_genero_relacion gr ON f.id_book = gr.id_libro
    JOIN libro_genero g ON gr.id_genero = g.id_genre
    WHERE g.id_genre = id_genero;
    RETURN stock_total;
END $$
DELIMITER ;
```

Función PRECIO_PROMEDIO

Descripción: esta función se utiliza para calcular el precio promedio de un libro específico, tomando todas las publicaciones que utilizan el

mismo libro en cuestión. Esta función combina datos clave de la tabla publicación para determinar el precio promedio de un libro particular basado en su ID.

Objetivo: el objetivo principal de esta función es proporcionar una manera eficiente de calcular el precio promedio de un libro específico. Al realizar este cálculo, la función permite a los usuarios obtener rápidamente una idea clara del precio medio al que se vende un libro en particular.


Tablas/Datos: esta función se compone de los siguientes elementos clave:

- Parámetro de entrada: libro_id - Un entero que representa el ID del libro para el cual se desea calcular el precio promedio.
- Variable local: prom - Un valor de punto flotante que almacena el precio promedio del libro especificado.
- Consulta SQL: La función utiliza una consulta SQL que calcula el promedio de precios de las publicaciones asociadas con el libro específico basado en el parámetro de entrada libro_id.

Esta función opera principalmente en la siguiente tabla:

- publicación: Contiene información detallada sobre las publicaciones de libros en el sistema, incluidos detalles específicos como el precio y el stock de los libros

Script de creación:



```
DELIMITER $$
CREATE FUNCTION precio_promedio(libro_id INT)
RETURNS FLOAT
DETERMINISTIC
BEGIN
    DECLARE prom FLOAT;
    SELECT AVG(p.price) INTO prom
    FROM publicacion p
    WHERE p.book = libro_id;
    RETURN prom;
END $$
DELIMITER ;
```

LISTADO DE STORED PROCEDURES

Stored Procedure SP_ORDENAR_TABLA

Descripción: este stored procedure se utiliza para ordenar dinámicamente una tabla específica de acuerdo con un campo específico. Este procedimiento almacenado acepta dos parámetros: el nombre de la tabla que se va a ordenar y el nombre del campo según el cual se realizará el ordenamiento.

Objetivo: el propósito principal de este stored procedure es proporcionar una forma dinámica de ordenar cualquier tabla en la base de datos según un campo específico, lo que facilita la organización y el análisis de los datos de la tabla en cuestión.

Tablas/Datos: este stored procedure se compone de los siguientes elementos clave:

- Parámetros de entrada:
 - nombre_tabla: Un parámetro de tipo VARCHAR que representa el nombre de la tabla que se va a ordenar.
 - orden_campo: Un parámetro de tipo VARCHAR que representa el campo según el cual se ordenará la tabla.
- Variable local: @q - Una variable que almacena la consulta SQL dinámica para ordenar la tabla según el campo especificado.

Script de creación:

```
DELIMITER $$
CREATE PROCEDURE sp_ordenar_tabla(IN nombre_tabla VARCHAR(50), IN orden_campo
    VARCHAR(50))
BEGIN
    SET @q = CONCAT('SELECT * FROM ', nombre_tabla, ' ORDER BY ', orden_campo);
    PREPARE stmt FROM @q;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END$$
DELIMITER ;
```

Stored Procedure SP_ELIMINAR_ENVIO_HASTA_FECHA

Descripción: este stored procedure se utiliza para eliminar todos los registros de la tabla envío que sean anteriores o iguales a la fecha ingresada. Este stored procedure acepta un parámetro de tipo DATETIME que representa la fecha límite hasta la cual se eliminarán los registros de envío antiguos.

Objetivo: el propósito principal de este stored procedure es proporcionar una forma rápida y eficiente de eliminar todos los registros de envío antiguos de la tabla envío en la base de datos hasta una fecha específica proporcionada como parámetro.

Tablas/Datos: este stored procedure se compone de los siguientes elementos clave:

- Parámetro de entrada: fecha - Un parámetro de tipo DATETIME que representa la fecha límite hasta la cual se eliminarán los registros de envío antiguos.
- Estructura condicional IF: El stored procedure utiliza una estructura condicional para verificar si la fecha proporcionada es anterior o igual a la fecha actual. Si esta condición se cumple, se ejecuta la operación de eliminación (esto se utiliza para proporcionar una capa de seguridad y evitar que se eliminen envíos futuros que serán los envíos pendientes que todavía no se hayan concretado).
- Comando DELETE: El stored procedure utiliza un comando DELETE para eliminar todos los registros de la tabla envío que tengan una fecha de envío anterior o igual a la fecha proporcionada.

Script de creación:

```
DELIMITER $$
CREATE PROCEDURE sp_eliminar_envio_hasta_fecha(IN fecha DATETIME)
BEGIN
    IF fecha <= CURRENT_TIMESTAMP() THEN
        DELETE FROM envio
            WHERE date_of_shipping <= fecha;
    END IF;
END$$
DELIMITER ;
```

LISTADO DE TRIGGERS Y TABLAS DE AUDITORÍA

Tabla AUD_USUARIO

AUD_USUARIO			
id_audit INT PK	id_user INT UNIQUE	user VARCHAR(255) NOT NULL	date DATETIME NOT NULL

Descripción: esta tabla es utilizada para llevar un registro de todos los usuarios que se van registrando en el E-commerce.

Listado de campos:

- id_audit INT PK AUTO_INCREMENT NOT NULL: este campo representa el "ID de cada registro de la tabla".
- id_user INT UNIQUE: este campo representa el "ID del usuario".
- user VARCHAR(255) NOT NULL: este campo representa el usuario que realizó el registro del usuario en la base de datos.
- date DATETIME NOT NULL: este campo representa la fecha en la que el usuario fue registrado.

Script de creación:

```
CREATE TABLE AUD_USUARIO(  
    id_audit INT AUTO_INCREMENT PRIMARY KEY,  
    id_user INT UNIQUE,  
    user VARCHAR(255) NOT NULL,  
    date DATETIME NOT NULL  
);
```

Tabla AUD_PUBLICACION

AUD_PUBLICACION			
id_audit INT PK	id_publication INT UNIQUE	user VARCHAR(255) NOT NULL	date DATETIME NOT NULL

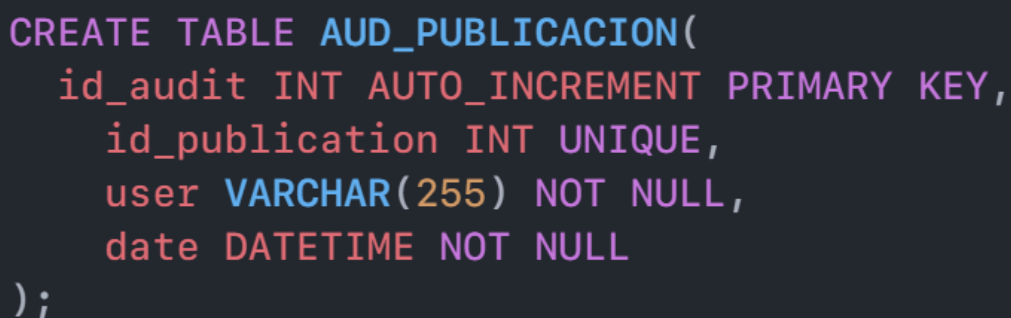
Descripción: esta tabla es utilizada para llevar un registro de todas las publicaciones junto con su usuario que se van realizando en el E-commerce.

Listado de campos:

- id_audit INT PK AUTO_INCREMENT NOT NULL: este campo representa el "ID de cada registro de la tabla".
- id_publication INT UNIQUE: este campo representa el "ID de la publicación".

- user VARCHAR(255) NOT NULL: este campo representa el usuario que realizó el registro de la publicación en la base de datos.
- date DATETIME NOT NULL: este campo representa la fecha en la que la publicación fue creada.

Script de creación:



```
CREATE TABLE AUD_PUBLICACION(  
    id_audit INT AUTO_INCREMENT PRIMARY KEY,  
    id_publication INT UNIQUE,  
    user VARCHAR(255) NOT NULL,  
    date DATETIME NOT NULL  
);
```

Trigger TR_CONTRASENA_SEGURA

Descripción: este trigger sirve para generar el encriptado de la contraseña del usuario al momento de insertar el registro.

Tipo: BEFORE INSERT

Tabla del trigger: USUARIO

Tabla(s) afectada(s): USUARIO

Script de creación:

```
DELIMITER //
```

```
CREATE TRIGGER tr_contrasena_segura
```

```
BEFORE INSERT ON usuario
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE encrypted_password VARCHAR(255);
```

```
    SET encrypted_password = SHA2(NEW.user_password, 256);
```

```
    SET NEW.user_password = encrypted_password;
```

```
END//
```

```
DELIMITER ;
```

Trigger TR_NUEVO_USUARIO

Descripción: este trigger sirve para insertar los registros de los usuarios que se van registrando, en una tabla de auditoría.

Tipo: AFTER INSERT

Tabla del trigger: USUARIO

Tabla(s) afectada(s): AUD_USUARIO

Script de creación:

```
CREATE TRIGGER tr_nuevo_usuario
AFTER INSERT ON usuario
FOR EACH ROW
INSERT INTO AUD_USUARIO
VALUES (NULL, NEW.id_user, SESSION_USER(), CURRENT_TIMESTAMP());
```

Trigger TR_VALIDACIONES_PUBLICACION

Descripción: este trigger sirve para realizar validaciones adicionales al momento de insertar una publicación, tales como control de cantidad de stock y de precios

Tipo: BEFORE INSERT

Tabla del trigger: PUBLICACION

Tabla(s) afectada(s): PUBLICACION

Script de creación:

```
DELIMITER //
```

```
CREATE TRIGGER tr_validaciones_publicacion
```

```
BEFORE INSERT ON publicacion
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    IF NEW.stock <= 0 THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
```

```
'El stock no puede ser un valor negativo o cero';
```

```
    END IF;
```

```
    IF NEW.price <= 0 THEN
```

```
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El precio no
```

```
puede ser un valor negativo o cero';
```

```
    END IF;
```

```
END//
```

```
DELIMITER ;
```

Trigger TR_NUEVA_PUBLICACION

Descripción: este trigger sirve para insertar los registros de las publicaciones que se van realizando, en una tabla de auditoría

Tipo: AFTER INSERT

Tabla del trigger: PUBLICACION

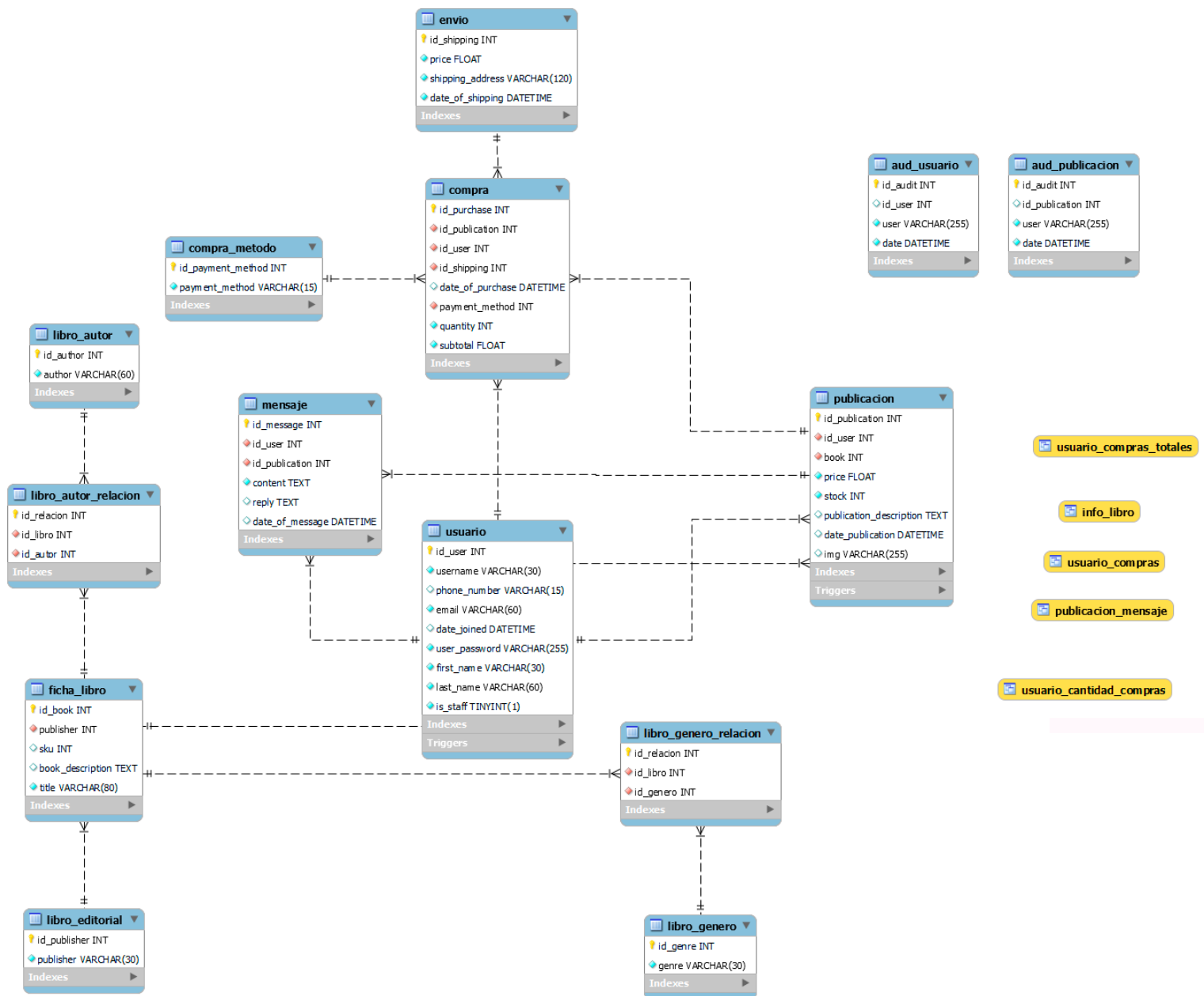
Tabla(s) afectada(s): AUD_PUBLICACION

Script de creación:



```
CREATE TRIGGER tr_nueva_publicacion  
AFTER INSERT ON publicacion  
FOR EACH ROW  
INSERT INTO AUD_PUBLICACION  
VALUES(NULL, NEW.id_publicacion, SESSION_USER(),  
CURRENT_TIMESTAMP());
```


REVERSE ENGINEER



Para acceder al EER en una mayor escala: [en el siguiente link](#)

SCRIPTS

La creacion de la base de datos junto con la insercion de datos se encuentra sintetizado [en un mismo archivo](#). Tambien se desgloso el mismo archivo en dos partes:

- [Script de creacion de la base de datos](#)
- [Script de insercion de datos](#)

El archivo de casos de prueba y ejemplos es el siguiente:

- [Script de pruebas y casos de uso](#)

TECNOLOGIAS UTILIZADAS

- [MySQL Workbench 8.0](#)
- [Drawio \(Diagrams\)](#)
- [Mockaroo \(Website\)](#)
- [Snapcode \(VSCode Extension\)](#)
- [Microsoft Word](#)
- [Git](#)
- [Github](#)