Base de Datos Multimedia 051

Autor: Nombre\_equipo/ Ecommerce

Fecha de creación: 7/09/2023

Última actualización:

Versión: 1.0

Integrantes: 3

-Carlos Aaron Valdes Hernandez

-Prado Tirado Angel Dario

-Emilio Gabriel Ramirez Leos

**Aprobaciones:**

|  |  |
| --- | --- |
| Aprobación 1 |  |
|  |  |
|  |  |
| Aprobación 2 |  |
|  |  |
|  |  |
| Aprobación 3 |  |

Tabla de Contenido

**1** **CONTROL DE DOCUMENTO 2**

**2** **NOMENCLATURA 3**

2.1 Database Schema 4

2.2 Database Objects 4

2.3 Concurrent Programs 6

2.4 Writing standard 6

*2.4.1* *SQL Coding Guidelines 6*

2.5 Error 7

2.6 Validations

2.7 Obstacles

**3** **COMMENTS 8**

*3.1.1* *Main Comment section 8*

# Control de Documento

**Récord de Cambios**

| **Fecha** | **Autor** | **Versión** | **Comentarios** |
| --- | --- | --- | --- |
|  |  |  |  |
| 10-sep-2023 | Dario, Carlos y Emilio | 1 | Esquema y objetos para la base |
| 20-oct-2023 | Dario, Carlos y Emilio | 2 | Login y conexión a la base |
| 24-nov-2023 | Dario,Carlos y Emilio | 3 | Funcionalidad Avanzada |
|  |  |  |  |

# Nomenclatura

## Database Schema

¿Qué esquema es el que voy a usar?

El esquema que se usará será BASE\_BDM este contendrá todos los objetos y es el que los usuarios podrán manejar.

Descripción del schema: La base de datos cuenta con un schema de 12 objetos de cada tabla.

## Database Objects

| Tipo de Objetos | Nombre de Formato | Explicación | Comentarios |
| --- | --- | --- | --- |
| Tables  -USUARIO  -ROL\_USUARIO  -DIRECCION\_USER  -LISTAS  -PRODUCTOS  -IMÁGENES  -CATEGORÍA  -VALORACIÓN  -METODO\_PAGO  -ORDEN\_COMPRA  -PRODUCTOS\_ORD  -FACTURA  -LISTA\_FORANEA  -IMAGENES\_FORANEA  -Messages  -Orden\_compra2 | -USUARIO  -ROL\_USUARIO  -DIRECCION\_USER  -LISTAS  -PRODUCTOS  -IMÁGENES  -CATEGORÍA  -VALORACIÓN  -METODO\_PAGO  -ORDEN\_COMPRA  -PRODUCTOS\_ORD  -FACTURA  -LISTA\_FORANEA  -IMAGENES\_FORANEA  -Mesages  -orden\_compra2 | USUARIO = Tabla para info de usuario.  ROL\_USUARIO = tabla para definir roles.  DIRECCION\_USER = tabla para guardar direcciones.  LISTAS = tabla para crear listas de productos.  PRODUCTOS = tabla para info de productos.  IMÁGENES = tabla para guardar imágenes.  CATEGORÍA = tabla para crear categorias.  VALORACIÓN = tabla para vincular valoraciones a productos.  METODO\_PAGO = tabla para guardar métodos de pago.  ORDEN\_COMPRA = tabla para guardar ordenes para compras.  PRODUCTOS\_ORD = tabla foránea para vincular productos con orden de compra.  FACTURA = tabla para guardar las facturas de compras.  LISTA\_FORANEA= sirve para evitar datos repetitivos en listas de productos.  IMAGENES\_FORANEA= sirve para evitar datos repetitivos en los productos  Message = Guarda y vincula los mensajes entre usuarios.  orden\_compra2= guarda los productos de el carrito y calcula el total y subtotal. | Se agregaron Nuevos elementos a la base de datos |
| Views | VIEW\_OBTENER\_LISTAS      VIEW\_OBTENER\_PRODUCTO        VIEW\_OBTENER\_CATEGORIAS | VIEW\_OBTENER\_LISTAS= obtiene las listas de un usuario en especifico.    VIEW\_OBTENER\_PRODUCTO= obtiene todos los productos.      VIEW\_OBTENER\_CATEGORIAS= obtiene todas las categorias. | CREACIÓN DE DOCUMENTO |
| Indexes | ID\_CATEGORIA      ID\_DIRECCION\_USUARIO.      ID\_FACTURA      ID\_IMAGENES        ID LISTA        ID\_LISTA\_FORANEA            msg\_id      ID\_ORDEN\_COMPRA2        ID\_PRODUCTO        IR\_DOL      ID\_USUARIO | table = CATEGORIA  x: U = unique.  table = DIRECCION\_USUARIO  x: U = unique.  table = FACTURA  x: U = unique  table = IMAGENES  x: U = unique  table = LISTA  x: U = unique    table =LISTA\_FORANEA  x: U = unique  table = Messages  x: U = unique  table = ORDEN\_COMPRA2  x: U = unique  table = PRODUCTOS  x: U = unique  table = ROL\_USUARIO  x: U = unique  table = USUARIOS  x: U = unique | SE AGREGARON ÍNDICES DE NUEVAS TABLAS |
| Sequences | FLxxx\_dddddddd\_SQ | Dddddddd = sequence name | Total length should not exceed 30 characters (example: FLPO\_POHEADER\_SQ |
| Procedures | PROC\_ACTUALIZAR\_PERFILES.  PROC\_ACTUALIZAR\_PRODCUTO  PROC\_AGREGAR\_A\_LISTA  PROC\_OBTENER\_ID\_USER  PROC\_REGISTRO\_USER  PROC\_ | PROC\_ACTUALIZAR\_PERFILES= actualiza la informacion de los perfiles de los usuarios.      PROC\_ACTUALIZAR\_PRODCUTO=actualiza la informacion de algun producto seleccionado.  PROC\_AGREGAR\_A\_LISTA= vincula un producto con una lista existente.  PROC\_OBTENER\_ID\_USER=obtiene el id de el usuario por medio de su correo.  PROC\_REGISTRO\_USER=registra los datos de un usuario. | CRECACION DE DOCUMENTO |
| Packages | FLxxx\_dddddddd\_PKG | dddddddd = package name | Total length should not exceed 30 characters (example: FLINV\_TRANSACTIONS\_PKG) |
| Functions | FLxxx\_dddddddd\_FN | dddddddd = function name | Total length should not exceed 30 characters (example: FLINV\_FUNCTION\_FN) |
| Synonyms | FLxxx\_dddddddd | dddddddd = synonym name (should be same as original object) | Total length should not exceed 30 characters (example: FLINV\_SYNONYMS). Synonyms for non-FL customizations will also be named the same as the original object. |
| Triggers |  | table = base table  x: B = before  A = after  y: I = insert  U = update  D = delete  z: S = statement  R = row | Total length should not exceed 30 characters  Triggers are automatically prepended with the tablename (\*1)  (example:  FLPO\_PURCHORD\_BIUDS  FLPO\_PURCHORD\_BIS  FLPO\_PURCHORD\_BUS  FLPO\_PURCHORD\_BDS) |
| Constraints | FLxxx\_dddddddd\_cons  FLxxx\_dddddddd\_FK#  FLxxx\_dddddddd\_PK  FLxxx\_dddddddd\_CK# | dddddddd = constraint name  # = sequence number (1 - 9) | Total length should not exceed 30 characters (example: FLPO\_PURCHORD\_FK1) |

## 

## Concurrent Programs

Detalle de programas externos

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre de Archivo | Ruta/Nombre de Archivo | Explicación | Comentarios |
| HTML | index.html  carrito.html | Lenguaje con el que se construyeron las páginas  ENTRE OTROS | VARIOS ARCHIVOS .html SE CAMBIARON A .php Y SE AGREGARON MAS ELEMENTOS HTML. |
| CSS | estilo\_carritro.css  estilo\_login.css | código con el que se le dio estilo a las páginas  .  ENTRE OTROS | SE AGREGARON MAS ELEMENTOS. |
| JAVASCRIPT | java\_log\_register.js  paginacion.js | código para funcionalidad de páginas  ENTRE OTROS | SE AGREGARON MAS ELEMENTOS. |
| PHP | CONEXION.php  LOGIN.php | sirve para manejo de backend de la página  ENTRE OTROS | SE AGREGARON MAS ELEMENTOS. |
|  |  |  |  |

## Writing standard

Descripción de scripts en caso de que existan más de 1 script que deban ejecutarse

### SQL Coding Guidelines



**SQL Statements:**

Sentencias de SQL que se necesitan y una pequeña explicación de cómo lo deben utilizar en el proyecto

**Sentencia 1: ACTUALIZAR USUARIO**

UPDATE usuario

SET CONTRASEÑA = 'nueva\_contraseña',

NOMBRE\_USUARIO = 'nuevo\_nombre\_usuario',

NOMBRE\_PERSONAL = 'nuevo\_nombre\_personal',

FECHA\_NACIMIENTO = 'nueva\_fecha\_nacimiento',

SEXO = 'nuevo\_sexo'

WHERE CORREO = 'correo\_usuario';

En este query, se actualizan varios campos de la tabla usuario con nuevos valores, y se hace la actualización basada en el correo electrónico (CORREO) que coincide con el valor proporcionado.

**Sentencia 2: ACTUALIZAR PRODUCTO**

UPDATE productos

SET NOMBRE = 'nuevo\_nombre',

DESCRIPCION = 'nueva\_descripcion',

FK\_CATEGORIA = 'nueva\_fk\_categoria',

PRECIO = 'nuevo\_precio',

CANTIDAD = 'nueva\_cantidad',

ESTADO = 'nuevo\_estado'

WHERE ID\_PRODUCTO = id\_deseado;

En este caso, la consulta actualiza los campos de la tabla productos con nuevos valores, y lo hace basado en el ID\_PRODUCTO que coincida con el valor proporcionado.

**Sentencia 3: VINCULAR LISTA Y PRODUCTO**

**INSERT INTO lista\_foranea**

**(ID\_LISTA, ID\_PRODUCTO)**

**VALUES ('valor\_fk\_categoria', 'valor\_id\_producto');**

**En esta consulta de inserción (INSERT), se agregan valores a la tabla lista\_foranea.**

**Sentencia 4: OBTENER PRODUCTO**

**SELECT NOMBRE, DESCRIPCION, PRECIO, VALORACION, CANTIDAD, FK\_CATEGORIA, VIDEO FROM productos**

**WHERE ID\_PRODUCTO = 'valor\_id';**

En esta consulta SELECT, se recuperan varios campos específicos de la tabla productos donde el ID\_PRODUCTO coincide con un valor específico.

**Sentencia 5: OBTENER CATEGORIA**

SELECT NOMBRE FROM categoria WHERE ID\_CATEGORIA = 'valor\_categoria';

En esta consulta SELECT, se está recuperando el campo NOMBRE de la tabla categoria donde el ID\_CATEGORIA coincide con un valor específico.

**Sentencia 6: OBTENER LISTAS**

SELECT \* FROM lista WHERE ID\_USUARIO = 'valor\_id\_usuario';

Esta consulta SELECT recupera todas las columnas (\* indica todas las columnas) de la tabla lista donde el campo ID\_USUARIO coincide con un valor específico.

**Sentencia 7: REGISTRAR USUARIO**

INSERT INTO usuario

(CORREO, CONTRASEÑA,

NOMBRE\_USUARIO,

NOMBRE\_PERSONAL,

FECHA\_NACIMIENTO,

SEXO,

IMAGEN,

ROL,

unique\_id) VALUES ('valor\_email', 'valor\_password', 'valor\_username', 'valor\_nombre', 'valor\_fecha\_nacimiento', 'valor\_sexo', 'valor\_rutaArchivo', valor\_rol, 'valor\_ran\_id');

Esta consulta INSERT añade una nueva fila a la tabla usuario con valores específicos en cada columna.

**Sentencia 8: RECUPERAR LISTAS**

**SELECT ID\_LISTA,**

**NOMBRE\_LISTA,**

**DESCRIPCION**

**FROM lista WHERE ID\_USUARIO = valor\_id;**

**En esta consulta SELECT, se recuperan las columnas ID\_LISTA, NOMBRE\_LISTA y DESCRIPCIÓN de la tabla lista donde el campo ID\_USUARIO coincide con un valor específico.**

**Sentencia 9: RECUPERAR PRODUCTO DE LISTA**

**SELECT**

**P.ID\_Producto,**

**P.Nombre AS NOMBRE,**

**P.Descripcion AS DESCRIPCION\_PRODUCTO,**

**P.Precio AS PRECIO\_PROD, L.ID\_LISTA, L.NOMBRE\_LISTA AS NOMBRE\_LISTA, L.Descripcion**

**AS DESCRIPCION FROM Productos P**

**JOIN lista\_foranea LF ON P.ID\_Producto = LF.ID\_Producto**

**JOIN lista L ON LF.ID\_LISTA = L.ID\_LISTA WHERE L.ID\_LISTA = valor\_id;**

**Esta consulta SELECT une tres tablas (Productos, lista\_foranea, y lista) utilizando JOIN y selecciona varias columnas de esas tablas. Se filtra el resultado por el campo ID\_LISTA de la tabla lista.**

**Sentencia 10: RECUPERAR PRODUCTOS DE UN USUARIO**

**SELECT**

**ID\_PRODUCTO, NOMBRE, DESCRIPCION FROM productos**

**WHERE ID\_USUARIO = valor\_id;**

**En esta consulta SELECT, se recuperan las columnas ID\_PRODUCTO, NOMBRE y DESCRIPCION de la tabla productos donde el campo ID\_USUARIO coincide con un valor específico.**

**Sentencia 11: RECUPERAR INFORMACION DE PRODUCTOS**

**SELECT ID\_PRODUCTO, NOMBRE, DESCRIPCION, PRECIO, VALORACION**

**FROM productos;**

**Esta consulta SELECT recupera las columnas ID\_PRODUCTO, NOMBRE, DESCRIPCION, PRECIO y VALORACION de la tabla productos.**

**Sentencia 12: REGISTRO LISTA**

**INSERT INTO lista (NOMBRE\_LISTA, DESCRIPCION, ESTADO, ID\_USUARIO) VALUES ('valor\_nombrelista', 'valor\_descripcion', 'valor\_priv', valor\_id);**

**Esta consulta INSERT añade una nueva fila a la tabla lista con valores específicos en cada columna.**

**Sentencia 13: REGISTRO CATEGORIA**

**INSERT INTO categoria (NOMBRE, ID\_USUARIO)**

**VALUES ('valor\_nombrecat', valor\_id);**

**Esta consulta INSERT agrega una nueva fila a la tabla categoria con valores específicos en las columnas NOMBRE e ID\_USUARIO.**

**Sentencia 14: REGISTRO DIRECCION**

**INSERT INTO direccion\_usuario**

**(DIRECCION, PAIS, CIUDAD, ESTADO, ID\_USUARIO)**

**VALUES ('$direccion', '$pais', '$ciudad', '$estado', valor\_id);**

**Esta consulta INSERT añade una nueva fila a la tabla direccion\_usuario con valores específicos en las columnas DIRECCION, PAIS, CIUDAD, ESTADO e ID\_USUARIO.**

**Sentencia 15: REGISTRO PRODUCTO**

**INSERT INTO productos**

**(NOMBRE, DESCRIPCION,**

**FK\_IMAGENES,**

**VIDEO, FK\_CATEGORIA,**

**PRECIO, CANTIDAD,**

**VALORACION, ESTADO,**

**CANTIDAD\_VENDIDOS,**

**FK\_COMENTARIOS,**

**ID\_USUARIO)**

**VALUES ('$nombre', '$descripcion', NULL, '$rutaArchivo', '$fk\_categoria', '$precio', '$cantidad', '$valoracion', '$estado', '$cantidad\_vendidos', NULL, valor\_id);**

**Esta consulta INSERT añade una nueva fila a la tabla productos con valores específicos en cada columna.**

## Error

## Errores detectables en el código

|  |  |  |
| --- | --- | --- |
| Nombre del Error | Explicacion | Comentarios |
| Pagina\_perdida | Login incorrecto O falta de sesion iniciada | Error visual, al momento de hacer login y no ser satisfactorio, este se redirecciona a una ventana vacia con un mensaje “Usuario no encontrado”. |
| Sin\_resultados\_por precios | No acepta datos numéricos solo letras | Error en la parte de búsqueda avanzada, la búsqueda no acepta valores numéricos para buscar por precio ya sea minimo o maximo |
| Program | xxx = application nameddd = The description of the program, | All custom program naming should start with Custom. IE, for AP, it could be ‘Custom AP: Build payments’. For PO, it could be ‘Custom PO: Summary GL Reconciliation Report’, etc. |
| Short Name | ddd = description | Found on Concurrent Programs dialog. Should match Executable and Execution File Name. |
| Description | xxx = application nameddd = The description of the program, | All custom program naming should start with Custom. IE, for AP, it could be ‘Custom AP: Build payments’. For PO, it could be ‘Custom PO: Summary GL Reconciliation Report’, etc. |

## Validations

Validation can encompass various checks:

**Data Type Validation:** Verifying that the input data is of the expected type (e.g., string, integer, date) before using it in operations or storing it.

**Format Validation:** Checking if the input data adheres to a specified format (e.g., email addresses, phone numbers, zip codes) using regular expressions or other validation techniques.

**Range Validation:** Ensuring that numerical input falls within an acceptable range (e.g., ensuring a user's age is between 18 and 100).

**Presence Validation:** Verifying that required fields or inputs are not empty or null.

**Consistency Validation:** Checking that the input data is consistent with other related data (e.g., validating a start date is before an end date).

**Cross-Field Validation:** Verifying relationships or constraints between multiple fields (e.g., ensuring the end date is after the start date).

Most commonly used validations would be password, username and email validations in order to allow certain characters and demand the usage of a specific format such as “aStringOfCharacters@provider.domain” at least.

Another typically used validation would be the one required for the password usually allowing the user to only register certain characters, a certain amount either by minimum or maximum amount of characters.

you can add validations in MySQL to ensure data integrity and enforce certain rules on your data. Here are a few ways to do this:

**Using Constraints:**

**Primary Key Constraint:** Ensures uniqueness in a column or a set of columns. Example: CREATE TABLE table\_name (id INT PRIMARY KEY, name VARCHAR(50));

**Unique Constraint:** Ensures that all values in a column or a set of columns are unique. Example: CREATE TABLE table\_name (email VARCHAR(50) UNIQUE, name VARCHAR(50));

**Foreign Key Constraint:** Maintains referential integrity between two tables.

**Using Triggers:**

**BEFORE INSERT and BEFORE UPDATE Triggers:** You can create triggers to execute logic before inserting or updating data. Within these triggers, you can include conditions and validations.

**Using Stored Procedures and Functions:**

You can encapsulate validation logic within stored procedures or functions to perform complex validations before inserting or updating data.

**Application Level Validation:**

While MySQL offers various validation techniques, sometimes more complex or context-specific validations are better suited for implementation within the application logic.

## Obstacles

## Developing a web page with a database can involve various challenges. Here are some common obstacles developers often encounter:

## **Database Design:** Poorly designed databases can lead to inefficiencies in data retrieval, storage, and management. Planning a database structure that fits the application's needs while considering scalability and performance is crucial.

## **Data Security:** Protecting sensitive user information is paramount. Ensuring that the database is secure from unauthorized access, SQL injection attacks, and other vulnerabilities requires robust security measures.

## **Performance Optimization:** Large datasets and complex queries can slow down web pages. Optimizing database queries, indexing frequently used columns, and caching strategies are necessary for better performance.

## **Compatibility and Integration:** Ensuring compatibility between different components of the web page (front-end, back-end, database) and integrating them seamlessly can be challenging, especially when dealing with various technologies and frameworks.

## **Scalability:** As the user base grows, scalability becomes crucial. Designing a database and web page architecture that can handle increased traffic and data volume without sacrificing performance is essential.

## **Data Integrity:** Maintaining data consistency and integrity, especially in scenarios involving concurrent user interactions or frequent updates, requires careful handling to prevent issues like data corruption or inconsistencies.

## **Version Control and Collaboration:** When multiple developers work on the same project, managing database schema changes, migrations, and version control while avoiding conflicts can be complex.

## **Testing and Debugging:** Ensuring that the web page and database work together seamlessly requires thorough testing. Identifying and fixing bugs, as well as handling exceptions and errors gracefully, is crucial for a smooth user experience.

## **Data Migration and Upgrades**: Upgrading the database or migrating data to a new system while minimizing downtime and data loss can be challenging and requires meticulous planning.

## **Documentation and Maintenance:** Keeping comprehensive documentation of the database structure, schemas, and relationships is essential for future maintenance and updates.

## 

# Comments

This comments section applies to all objects changes or creation. This includes creation scripts, reports, forms, PL/SQL scripts, etc. It is very important to have descriptive comments to describe the purpose of the functions throughout the program. Each program need to contain a main comment section. Comments should be included to further describe programming steps/sections as needed. Comments should be used to mark the begin and the end of the changes made to the section of the code within a standard program. Comments should also include the Fix/issue/defect number reference where applicable.

### Main Comment section

The main comment section need to contain the following information:

|  |  |
| --- | --- |
| Program Name | The actual program name. Ecomerce |
| Description/Purpose | Un entorno dedicado a la venta de productos en linea, parecido a amazon, mercado libre u otros por el estilo. |
| Reference Number | This is the RICEF number or Coding Package number. This will allow tracking of multiple objects related to a RICEF or a Package. To allow for a quick search, the reference number should be prefixed by $FL$. IE, if the reference number is 12345 , then it should be coded as $FL$12345. |
| Change History | This is the change history of the program, The change history need to include the following:  Date Changes – The date the program was created/modified  Changed By – The programmer name who changed it.  Description – The description as to what has changed. |

Example:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Name : FLPO\_EMP\_APPRV\_PKG \*

\* Description : <Description on what the program does>

\*

\* Reference Number: $FL$XXXXX (where XXXXX can be RICEF number or a Package Number.

\*

\* Change History :

\* Date Modified/created by

\* -----------------------------------------------------------------

\* <Date of changes> <Programmer’s name>

\* <Change description>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Nomenclatura que deberá contener cada script o .SQL

La nomenclatura utilizada en los scripts de SQL puede variar según las preferencias personales o las convenciones de nomenclatura específicas de una organización, pero existen algunas convenciones comunes que son ampliamente aceptadas. Aquí hay algunos ejemplos de nomenclatura común en los scripts de SQL:

1. **Nombres de Tablas:**

- Suelen estar en singular.

- Se recomienda usar nombres descriptivos, por ejemplo, "users" en lugar de "tbl\_usr".

2. **Nombres de Columnas:**

- También suelen estar en singular.

- Nombres descriptivos que reflejen el contenido de la columna, como "nombre" en lugar de "col1".

3. **Nombres de Esquemas o Bases de Datos:**

- Usar un prefijo que indique la base de datos o el esquema al que pertenecen las tablas, como "dbo.tabla" o "public.tabla".

4. **Palabras clave y Funciones SQL:**

- Se suelen escribir en mayúsculas, como "SELECT", "FROM", "WHERE", "MAX", "SUM", etc.

5. **Nombres de Variables o Parámetros:**

- Suelen usar un formato que indique claramente que son variables o parámetros, como "@nombre\_variable" o ":nombre\_parametro".

6. **Alias de Tablas y Columnas:**

- Cuando se utilizan alias, es común utilizar letras minúsculas o abreviaciones, como "t" para una tabla y "c" para una columna.

7. **Comentarios:**

- Se utilizan para documentar el código y suelen comenzar con "--" en SQL. Por ejemplo: "-- Este es un comentario".

8. **Convenciones de Nombres Compuestas:**

- Para nombres compuestos, como "nombre\_completo," se pueden usar guiones bajos ("nombre\_completo") o notación CamelCase ("nombreCompleto").

Es importante seguir una convención de nomenclatura consistente en sus scripts de SQL para facilitar la lectura, el mantenimiento y la colaboración con otros desarrolladores. Además, algunas bases de datos, como PostgreSQL y MySQL, tienen reglas específicas para los nombres de objetos, por lo que es importante conocer y seguir las convenciones de nomenclatura de la base de datos que esté utilizando.

DATA DICTIONARY

Basado en el texto proporcionado y la estructura que sugieres en tu consulta anterior, aquí tienes un ejemplo de un diccionario de datos para algunas de las tablas mencionadas en el script SQL:

\*\*Diccionario de Datos\*\*

1. \*\*Tabla: `categoria`\*\*

- \*\*ID\_CATEGORIA\*\*: (int) Identificador único de la categoría.

- \*\*NOMBRE\*\*: (varchar) Nombre de la categoría.

- \*\*ID\_USUARIO\*\*: (int) Identificador del usuario al que pertenece la categoría.

2. \*\*Tabla: `direccion\_usuario`\*\*

- \*\*ID\_DIRECCION\_USUARIO\*\*: (int) Identificador único de la dirección de usuario.

- \*\*DIRECCION\*\*: (varchar) Dirección del usuario.

- \*\*PAIS\*\*: (varchar) País de la dirección.

- \*\*CIUDAD\*\*: (varchar) Ciudad de la dirección.

- \*\*ESTADO\*\*: (int) Identificador del estado de la dirección.

- \*\*ID\_USUARIO\*\*: (int) Identificador del usuario al que pertenece la dirección.

3. \*\*Tabla: `factura`\*\*

- \*\*ID\_FACTURA\*\*: (int) Identificador único de la factura.

- \*\*PRECIO\_SUBTOTAL\*\*: (float) Subtotal de la factura.

- \*\*PRECIO\_TOTAL\*\*: (float) Precio total de la factura.

- \*\*FECHA\*\*: (date) Fecha de la factura.

- \*\*ID\_METODO\_PAGO\*\*: (int) Identificador del método de pago.

- \*\*ID\_USUARIO\*\*: (int) Identificador del usuario al que pertenece la factura.

- \*\*ID\_ORDEN\_COMPRA\*\*: (int) Identificador de la orden de compra asociada a la factura.

4. \*\*Tabla: `imagenes`\*\*

- \*\*ID\_IMAGENES\*\*: (int) Identificador único de la imagen.

- \*\*RUTA\_IMAGEN\*\*: (varchar) Ruta de la imagen.

5. \*\*Tabla: `imagenes\_foraneas`\*\*

- \*\*ID\_IMAGEN\_FOR\*\*: (int) Identificador único de la imagen foránea.

- \*\*ID\_PRODUCTO\*\*: (int) Identificador del producto al que se refiere la imagen.

- \*\*ID\_IMAGEN\*\*: (int) Identificador de la imagen asociada.

6. \*\*Tabla: `lista`\*\*

- \*\*ID\_LISTA\*\*: (int) Identificador único de la lista.

- \*\*NOMBRE\_LISTA\*\*: (varchar) Nombre de la lista.

- \*\*DESCRIPCION\*\*: (varchar) Descripción de la lista.

- \*\*ESTADO\*\*: (int) Estado de la lista.

- \*\*ID\_USUARIO\*\*: (int) Identificador del usuario propietario de la lista.

7. \*\*Tabla: `lista\_foranea`\*\*

- \*\*ID\_LISTA\_FORANEA\*\*: (int) Identificador único de la lista foránea.

- \*\*ID\_LISTA\*\*: (int) Identificador de la lista a la que pertenece el producto.

- \*\*ID\_PRODUCTO\*\*: (int) Identificador del producto asociado a la lista.

8. \*\*Tabla: `metodo\_pago`\*\*

- \*\*ID\_METODO\_PAGO\*\*: (int) Identificador único del método de pago.

- \*\*NOMBRE\*\*: (varchar) Nombre del método de pago.

9. \*\*Tabla: `orden\_compra`\*\*

- \*\*ID\_ORDEN\_COMPRA\*\*: (int) Identificador único de la orden de compra.

- \*\*SUBTOTAL\*\*: (float) Subtotal de la orden de compra.

- \*\*TOTAL\*\*: (float) Total de la orden de compra.

10. \*\*Tabla: `productos`\*\*

- \*\*ID\_PRODUCTO\*\*: (int) Identificador único del producto.

- \*\*NOMBRE\*\*: (varchar) Nombre del producto.

- \*\*DESCRIPCION\*\*: (varchar) Descripción del producto.

- \*\*FK\_IMAGENES\*\*: (int) Identificador de la imagen asociada al producto.

- \*\*VIDEO\*\*: (varchar) Enlace al video relacionado con el producto.

- \*\*FK\_CATEGORIA\*\*: (int) Identificador de la categoría a la que pertenece el producto.

- \*\*PRECIO\*\*: (float) Precio del producto.

- \*\*CANTIDAD\*\*: (int) Cantidad disponible del producto.

- \*\*VALORACION\*\*: (varchar) Valoración del producto.

- \*\*ESTADO\*\*: (int) Estado del producto.

- \*\*CANTIDAD\_VENDIDOS\*\*: (int) Cantidad de unidades vendidas del producto.

- \*\*FK\_COMENTARIOS\*\*: (int) Identificador de los comentarios asociados al producto.

- \*\*ID\_USUARIO\*\*: (int) Identificador del usuario que publicó el producto.

11. \*\*Tabla: `productos\_ordenados`\*\*

- \*\*ID\_PRODUCTOS\_ORDENADOS\*\*: (int) Identificador único del registro de productos ordenados.

- \*\*ID\_PRODUCTO\*\*: (int) Identificador del producto incluido en la orden.

- \*\*ID\_ORDEN\_COMPRA\*\*: (int) Identificador de la orden de compra a la que pertenece el producto.

12. \*\*Tabla: `rol\_usuario`\*\*

- \*\*ID\_ROL\_USUARIO\*\*: (int) Identificador único del rol de usuario.

- \*\*DESCRIPCION\*\*: (varchar) Descripción del rol de usuario.

13. \*\*Tabla: `usuario`\*\*

- \*\*ID\_USUARIO\*\*: (int) Identificador único del usuario.

- \*\*CORREO\*\*: (varchar) Dirección de correo electrónico del usuario.

- \*\*CONTRASEÑA\*\*: (varchar) Contraseña del usuario.

- \*\*NOMBRE\_USUARIO\*\*: (varchar) Nombre de usuario en la plataforma.

- \*\*NOMBRE\_PERSONAL\*\*: (varchar) Nombre completo del usuario.

- \*\*FECHA\_NACIMIENTO\*\*: (date) Fecha de nacimiento del usuario.

- \*\*SEXO\*\*: (varchar) Género o sexo del usuario.

- \*\*ROL\*\*: (int) Identificador del rol de usuario.

14. \*\*Tabla: `messages`\*\*

- \*\*msg\_id\*\*: (int) Identificador único del mensaje.

- \*\*incomingmessage\_id\*\*: (int) Identificador único del mensaje enviado.

- \*\*outgoingmessage\_id\*\*: (int) Identificador único del mensaje enviado.

- \*\*msg\*\*: (var) cuerpo de el Mensage.

Captura de rediseño de algunas de las pantallas

Se hicieron cambios en la interfaz de las pantallas para mejorar la experiencia del usuario además de mantener un mismo diseño en el sitio web.

