



APLICACIÓN DE AHORROS



Sergio Ruiz Ortiz

Fecha:13/12/2025

Curso:1ºDAW

ÍNDICE

| | |
|-------------------------------------|----|
| ÍNDICE..... | 1 |
| DESCRIPCIÓN DEL PROYECTO..... | 2 |
| MODELO ENTIDAD RELACIÓN..... | 4 |
| MODELO RELACIONAL EN WORKBENCH..... | 5 |
| CAMBIOS CON SET Y ENUM..... | 5 |
| CONSULTAS SQL..... | 6 |
| CONCLUSIÓN..... | 12 |

DESCRIPCIÓN DEL PROYECTO

Este proyecto trata sobre una aplicación para gestionar el dinero personal, pensada para ayudar a las personas a organizar sus ahorros y controlar sus gastos de una forma sencilla e intuitiva. He elegido este proyecto porque considero importante organizar el dinero de forma clara y responsable.


Cada usuario puede registrarse en la aplicación y se guardan sus datos personales y de acceso, como su identificador, nombre, apellidos, fecha de nacimiento, nombre de usuario, contraseña, correo electrónico y el país en el que vive. El usuario es el elemento principal del sistema, ya que es quien realiza todas las acciones dentro de la aplicación.

Una vez registrado, el usuario puede crear distintas metas de ahorro, por ejemplo ahorrar para un viaje o para una compra importante. De cada meta se guarda su identificador, el nombre de la meta, el estado en el que se encuentra, la prioridad, la cantidad total que se desea ahorrar y la cantidad que se ha conseguido hasta el momento. Además, se registran las fechas de inicio y de fin de cada meta.

Para gestionar el dinero dentro de las metas de ahorro, el usuario puede realizar transacciones, que permiten añadir o retirar dinero. Cada transacción queda asociada a un usuario y a una meta concreta, y se guarda el número de transacción, la cantidad, el tipo de movimiento (ingreso o retirada) y la fecha en la que se realiza.

Además, el usuario puede crear distintas listas de presupuesto, que sirven para establecer límites de gasto. Por ejemplo, se puede definir un presupuesto para comida, ocio o compras. De cada lista se guarda su identificador, el nombre, una descripción y la cantidad máxima que se puede gastar .

Las listas de presupuesto se organizan mediante categorías, que permiten identificar fácilmente el tipo de gasto. De cada categoría se guarda su identificador, el nombre, una breve descripción y el color del icono que la representa. La aplicación también incluye un sistema de notificaciones, que informa al usuario sobre distintos eventos



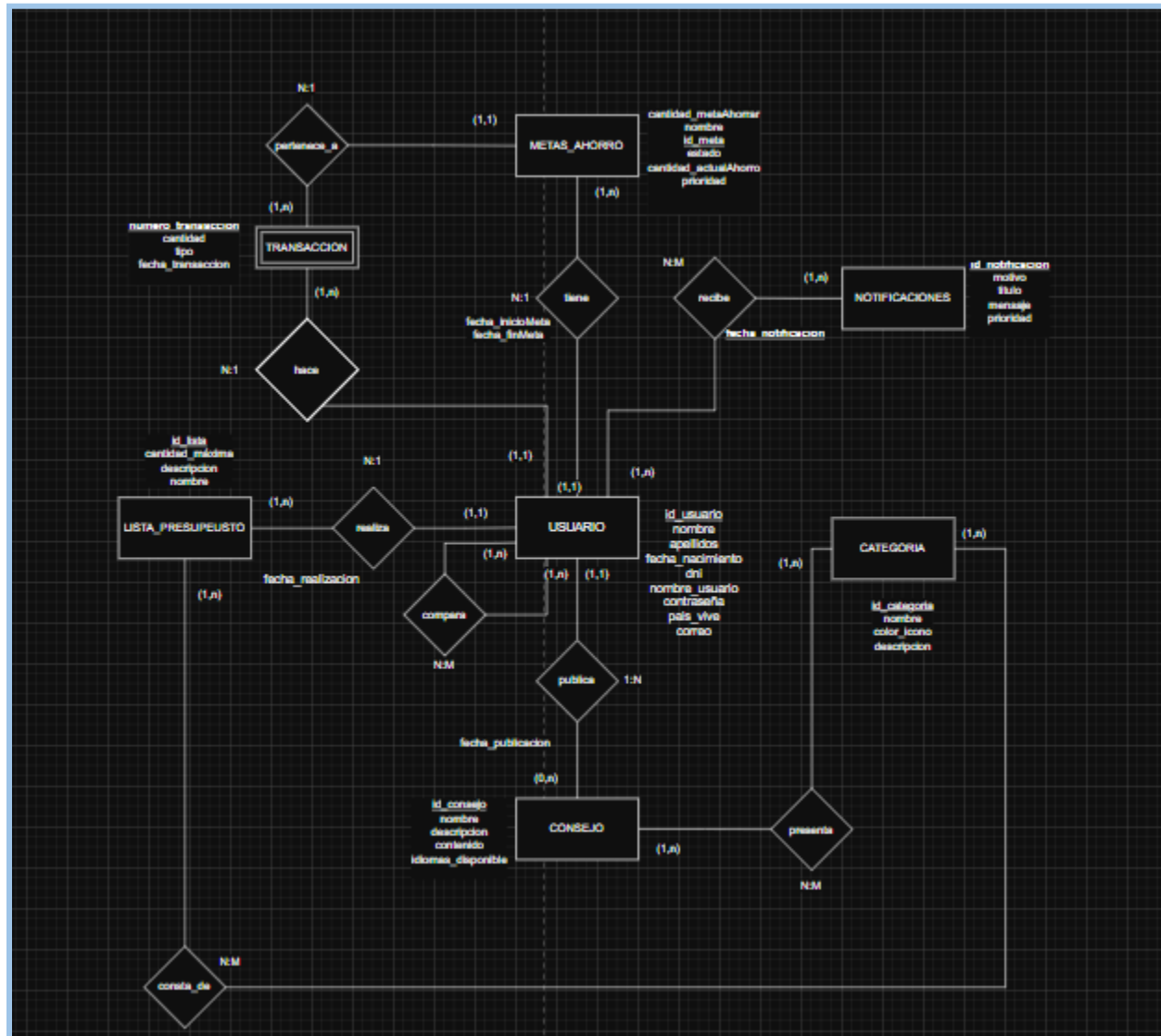
importantes, como el estado de una meta de ahorro o avisos relacionados con sus presupuestos.

De cada notificación se guarda su identificador, el motivo, el título, el mensaje y su prioridad, así como la fecha en la que se recibe.

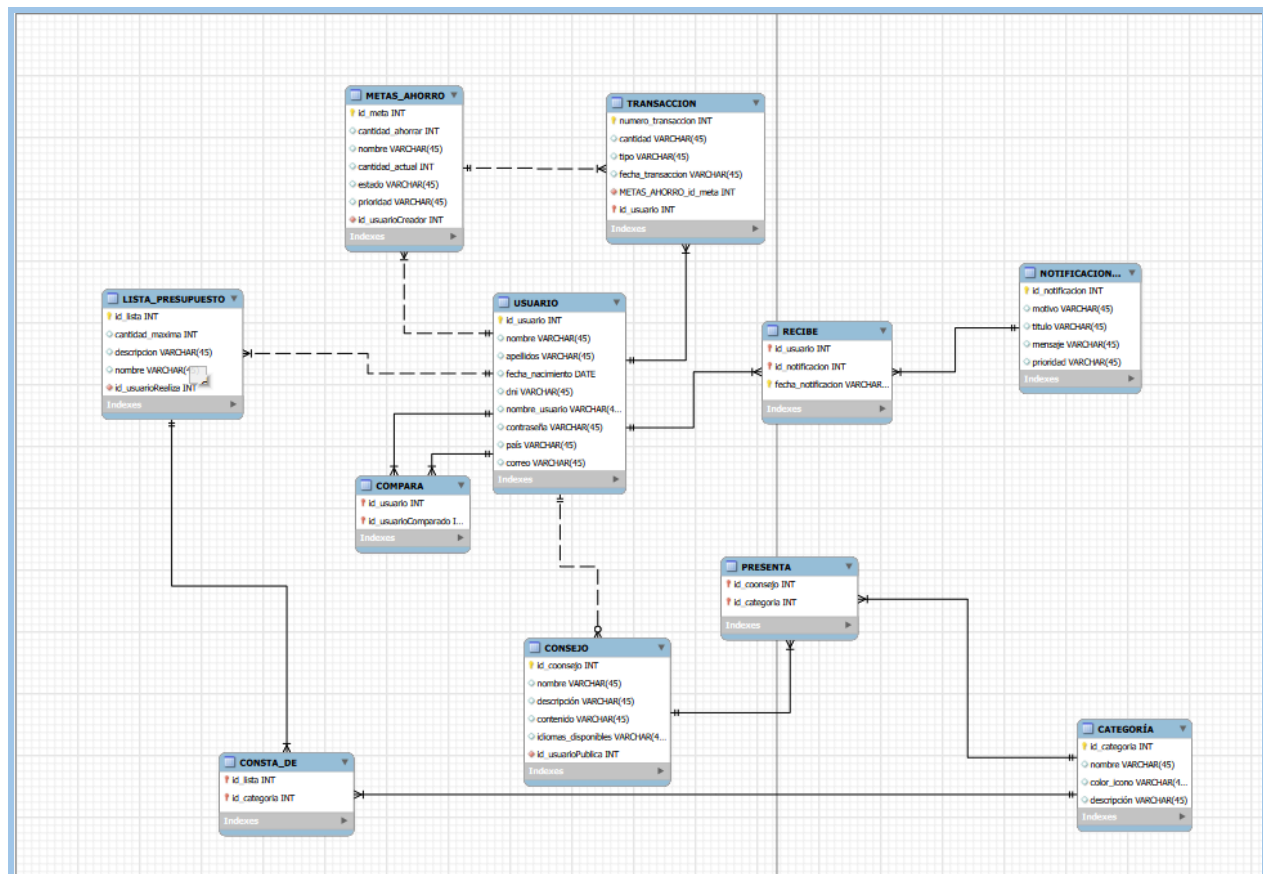
Por otro lado, los usuarios pueden publicar consejos relacionados con la gestión del dinero, con el objetivo de ayudar a otros usuarios. De cada consejo se guarda su identificador, el nombre, una descripción y el contenido, además de la fecha en la que se publica.

Por último, la aplicación permite que los usuarios se comparen entre sí, de manera que puedan ver su situación económica en relación con otros usuarios, fomentando un la competencia entre usuarios, lo que los va a llevar a tomárselo más en serio.

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL EN WORKBENCH



CAMBIOS CON SET Y ENUM

```
ALTER TABLE METAS_AHORRO MODIFY COLUMN estado ENUM('activo', 'pendiente', 'inactivo') NOT NULL DEFAULT 'inactivo';

ALTER TABLE METAS_AHORRO MODIFY COLUMN prioridad ENUM('baja', 'media', 'alta') not null default 'baja';

ALTER TABLE NOTIFICACIONES modify column prioridad ENUM('baja', 'media', 'alta') not null default 'baja';

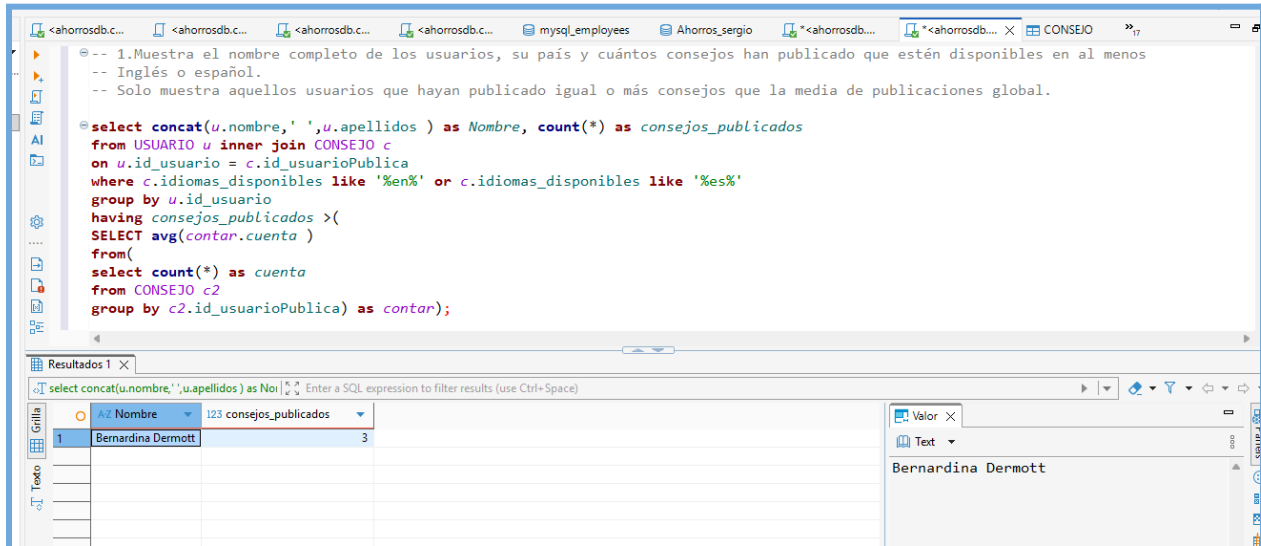
ALTER TABLE TRANSACCION modify column tipo ENUM('compra', 'venta', 'transferencia') not null default 'compra';

alter table PRESENTA CHANGE id_consejo id_consejo INT;

ALTER TABLE CONSEJO CHANGE id_consejo id_consejo INT;

ALTER TABLE CONSEJO
ADD COLUMN idiomas_disponibles SET('es', 'en', 'fr', 'pt', 'de', 'it')
NOT NULL DEFAULT 'es' AFTER contenido;
```

CONSULTAS SQL



The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
-- 1.Muestra el nombre completo de los usuarios, su país y cuántos consejos han publicado que estén disponibles en al menos
-- Inglés o español.
-- Solo muestra aquellos usuarios que hayan publicado igual o más consejos que la media de publicaciones global.

select concat(u.nombre,' ',u.apellidos ) as Nombre, count(*) as consejos_publicados
from USUARIO u inner join CONSEJO c
on u.id_usuario = c.id_usuarioPublica
where c.idiomas_disponibles like '%en%' or c.idiomas_disponibles like '%es%'
group by u.id_usuario
having consejos_publicados >(
SELECT avg(contar.cuenta )
from(
select count(*) as cuenta
from CONSEJO c2
group by c2.id_usuarioPublica) as contar);
```

The results pane shows the following table:

| | AZ Nombre | 123 consejos_publicados |
|---|--------------------|-------------------------|
| 1 | Bernardina Dermott | 3 |

The results pane also shows a text box with the value "Bernardina Dermott".

ahorrosdb.cpmf ... amazonaws.com Ahorros_sergio

<ahorrosdb.c... <ahorrosdb.c... <ahorrosdb.c... <ahorrosdb.c... mysql_employees Ahorros_sergio *<aho

```
-- 2. Para cada usuario, muestra su nombre de usuario (en mayúsculas) y
-- el nombre de su lista de presupuesto. Indica también el nombre de la categoría
-- asociada a esa lista. Solo muestra aquellas listas que
-- superen la media de gasto máximo.

select UPPER(u.nombre) as nombre, lp.nombre as nombre_lista, c.id_categoria
from USUARIO u inner join LISTA_PRESUPUESTO lp
on u.id_usuario = lp.id_usuarioRealiza
inner join CONSTA_DE cd
on lp.id_lista = cd.id_lista
inner join CATEGORIA c
on cd.id_categoria = c.id_categoria
where lp.cantidad_maxima > (
select avg(lp2.cantidad_maxima)
from LISTA_PRESUPUESTO lp2);
```

LISTA_PRESUPUESTO(+) 1 X

select UPPER(u.nombre) as nombre, lp.nombre as nombr | Enter a SQL expression to filter results (use Ctrl+Space)

| | AZ nombre | AZ nombre_lista | 123 id_categoria |
|----|------------|-----------------------|------------------|
| 1 | MILTY | Control Financiero | 3 |
| 2 | VIVYAN | Presupuesto Familiar | 4 |
| 3 | ARDRA | Presupuesto Básico | 5 |
| 4 | GUILLAUME | Resumen Financiero | 6 |
| 5 | BEVERLEY | Presupuesto General | 1 |
| 6 | LILLIE | Presupuesto Mensual | 3 |
| 7 | RUTHERFORD | Control Mensual | 4 |
| 8 | JANOT | Plan de Gastos | 6 |
| 9 | ANDI | Gastos Personales | 2 |
| 10 | VIRGIL | Presupuesto Personal | 3 |
| 11 | ELLI | Plan Financiero | 4 |
| 12 | EMMET | Presupuesto Familiar | 7 |
| 13 | CLARITA | Control Financiero | 1 |
| 14 | HARTLEY | Presupuesto Mensual | 2 |
| 15 | CONNOR | Resumen Financiero | 4 |
| 16 | WINSLOW | Control Mensual | 5 |
| 17 | PERKIN | Presupuesto del Hogar | 7 |
| 18 | QUINTUS | Mi Presupuesto | 3 |
| 19 | GERALD | Mi Presupuesto | 4 |
| 20 | TADIO | Presupuesto General | 1 |
| 21 | TARYN | Presupuesto General | 3 |
| 22 | ELVINA | Lista de Gastos | 4 |
| 23 | BRIA | Control Mensual | 6 |


```
-- 3. Calcula cuántos días han pasado desde que cada usuario recibió su última not
-- Muestra el nombre del usuario, la fecha de la última notificación y
-- los días transcurridos. Si nunca ha recibido una, usa IFNULL para poner "0".
create view consulta_3 as
select u.nombre as nombre_usuario,
ifnull(max(r.fecha_notificacion), 'no le ha llegado') as ultima_fecha,
ifnull(DATEDIFF(NOW(), r.fecha_notificacion), 'no tiene') as dias_diferencia
from USUARIO u left join RECIBE r
on u.id_usuario = r.id_usuario
group by u.id_usuario ;
-- 4. Compara lo que un usuario ha gastado de verdad con su presupuesto. Muestra e
-- su presupuesto máximo, el total de sus transacciones y su saldo restante.
create view consulta_4 as
select concat(left(u.apellidos, LENGTH(u.apellidos)-1), UPPER(right(u.apellidos, 1)
(1) cantidad maxima -(SELECT sum(2) cantidad )
```

SUARIO 1 X

select u.nombre as nombre_usuario, ifnull(max(r.fecha_n | Enter a SQL expression to filter results (use Ctrl+Space)

| | AZ nombre_usuario | AZ ultima_fecha | AZ dias_diferencia |
|---|-------------------|------------------|--------------------|
| 1 | Gabie | no le ha llegado | no tiene |
| 2 | Milty | no le ha llegado | no tiene |
| 3 | Vivyan | no le ha llegado | no tiene |
| 4 | Ardra | no le ha llegado | no tiene |

-- 4. Compara lo que un usuario ha gastado de verdad con su presupuesto. Muestra el apellido su presupuesto máximo, el total de sus transacciones y su saldo restante.

```

create view consulta_4 as
select concat(left(u.apellidos,LENGTH(u.apellidos)-1),UPPER(right(u.apellidos,1)))
,(lp.cantidad_maxima -(SELECT sum(t2.cantidad)
from USUARIO u2 inner join TRANSACCION t2
on u2.id_usuario = t2.id_usuario
where u.id_usuario = u2.id_usuario )) as comparación, lp.cantidad_maxima,
count(distinct t.numero_transaccion) as numero_transacciones
from TRANSACCION t INNER JOIN USUARIO u
ON t.id_usuario = u.id_usuario
inner join LISTA_PRESUPUESTO lp
on u.id_usuario = lp.id_usuarioRealiza
group by u.id_usuario;

```

-- 5. Muestran los usuarios cuyo gasto total en transacciones supera

TA_PRESUPUESTO 1 x

select concat(left(u.apellidos,LENGTH(u.apellidos)-1),UPPER(right(u.apellidos,1))) Enter a SQL expression to filter results (use Ctrl+Space)

| | AZ concat(left(u.apellidos,LENGTH(u.apellidos)-1),UPPER(right(u.apellidos,1))) | 123 comparación | 123 cantidad_maxima | 123 numero_transacciones |
|---|--|-----------------|---------------------|--------------------------|
| | SumptoN | 5.748,54 | 6.002,29 | 1 |
| | MulvaneY | 5.572,16 | 6.082,81 | 1 |
| | KynseY | 6.382,81 | 7.234,57 | 1 |
| | GentreaU | 6.998,58 | 7.242,45 | 1 |
| | De BeauchemP | 1.282,32 | 1.324,07 | 1 |
| | Suddell | 7.343,63 | 7.387,93 | 1 |
| | MinoR | 3.666,6 | 4.536,19 | 1 |
| | RhymeR | 6.672,79 | 6.831,45 | 1 |
| | McClintoN | 7.532,89 | 7.688,24 | 1 |
| 0 | BirkwoodD | -560,95 | 380,88 | 1 |
| 1 | JoselsoN | 4.984,25 | 5.003,91 | 1 |
| 2 | MellyY | 2.046,06 | 2.397,3 | 1 |
| 3 | MoxsoN | 1.426,01 | 1.739,22 | 1 |
| 4 | HilleY | 5.198,28 | 5.791,91 | 1 |
| 5 | FouracrE | 4.075,94 | 4.935,68 | 1 |
| 6 | Jell | 8.733,86 | 9.660,62 | 1 |
| 7 | ArthaN | 2.539,67 | 2.603,09 | 1 |
| 8 | HilleY | 3.523,84 | 3.700,24 | 1 |
| 9 | CorteiS | 8.388,24 | 9.117,75 | 1 |
| 0 | BritneR | 6.942,12 | 7.521,41 | 1 |
| 1 | HollingdalE | 9.645,67 | 9.991,5 | 1 |
| 2 | MitchensoN | 248,94 | 323,84 | 1 |
| 3 | Chappell | 6.844,05 | 7.124,76 | 1 |
| 4 | BriscoE | 8.339,19 | 8.765,38 | 1 |
| 5 | ConderD | 4.057,01 | 4.112,67 | 1 |

MySQL Workbench interface showing a SQL query and its results.

SQL Query:

```
-- 5. Mostrar los usuarios cuyo gasto total en transacciones supera
-- la media de gasto de todos los usuarios de la aplicación
select u.nombre, sum(t.cantidad) as suma
from USUARIO u inner join TRANSACCION t
on u.id_usuario = t.id_usuario
group by u.id_usuario
having suma > (
select avg(cantidad_emplado.cantidad)
from(
select sum(t2.cantidad) as cantidad
from USUARIO u2 inner join TRANSACCION t2
on u2.id_usuario = t2.id_usuario
group by u2.id_usuario) as cantidad_emplado);
```

Query Results:

| | A-Z nombre | 123 suma |
|----|-------------|----------|
| 1 | Vivyan | 510,65 |
| 2 | Ardra | 851,76 |
| 3 | Donovan | 869,59 |
| 4 | Mag | 941,83 |
| 5 | Andi | 593,63 |
| 6 | Virgil | 859,74 |
| 7 | Elli | 926,76 |
| 8 | Emmet | 729,51 |
| 9 | Clarita | 579,29 |
| 10 | Darleen | 887,19 |
| 11 | Karlotta | 936,49 |
| 12 | Quintus | 566,09 |
| 13 | Filide | 897,99 |
| 14 | Tadio | 562,59 |
| 15 | Taryn | 992,37 |
| 16 | Elvina | 966,27 |
| 17 | Debbie | 757,98 |
| 18 | Jacquenette | 606,68 |
| 19 | Celie | 633,83 |
| 20 | Daisie | 700,8 |
| 21 | Mehetabel | 718,93 |
| 22 | Lars | 749,63 |
| 23 | Lillian | 820,17 |
| 24 | Kimberlee | 863,42 |

-- 6. Encontrar usuarios que se han registrado pero no han creado ninguna lista de presupuesto. (usuarios que no usan la app).

```
select u.nombre
from USUARIO u left join LISTA_PRESUPUESTO lp
on u.id_usuario = lp.id_usuarioRealiza
where lp.id_lista is null;
```

USUARIO 1 X

```
select u.nombre from USUARIO u left join LISTA_PRESUPUESTO lp on u.id_usuario = lp.id_usuarioRealiza
```

AZ nombre

| 1 | Borja |
|---|-------|

Valor X

Text

Borja

-- 7. Encontrar usuarios que tienen metas de ahorro ambiciosas (más de 500€) pero que no aparecen en la tabla de transacciones. -- Es decir, tienen el deseo de ahorrar pero no han registrado ni un solo movimiento en la base de datos.

```
SELECT u.nombre as nombre
from USUARIO u
where u.id_usuario in(
select u3.id_usuario
from USUARIO u3
where u3.id_usuario not in(
select u4.id_usuario
from USUARIO u4 inner join TRANSACCION t
on u4.id_usuario = t.id_usuario)) and u.id_usuario in(
select u2.id_usuario
from USUARIO u2 inner join METAS_AHORRO ma
on u2.id_usuario = ma.id_usuarioCreador
where ma.cantidad_ahorrar >500);
```

USUARIO 1 X

```
SELECT u.nombre as nombre from USUARIO u where u.id_usuario in(
select u3.id_usuario
from USUARIO u3
where u3.id_usuario not in(
select u4.id_usuario
from USUARIO u4 inner join TRANSACCION t
on u4.id_usuario = t.id_usuario)) and u.id_usuario in(
select u2.id_usuario
from USUARIO u2 inner join METAS_AHORRO ma
on u2.id_usuario = ma.id_usuarioCreador
where ma.cantidad_ahorrar >500);
```

AZ nombre

| 1 | Gabie |
|---|-------|

Valor X

Text

Gabie



CONCLUSIÓN

Con este proyecto he aprendido lo importante que es darle el valor correcto a cada dato al definir las tablas; si te equivocas con el tipo, luego las consultas fallan o te truncan la información. También me ha servido para ver cómo se monta una base de datos real en AWS y cómo manejar un volumen de 1000 registros sin que se me escape nada. A futuro me gustaría mejorar la precisión a la hora de poner valor a los atributos, y me gustaría aumentar el nivel de mis consultas.



Enlace a GitHub: https://github.com/sergioruiz28282/Ahorros_ProyectoBD_SRO.git