

Consultoría de Bancos

Bases de Datos.....	2
Enunciado.....	2
Diseño conceptual.....	4
Diseño lógico.....	4
Entidades.....	4
Relaciones N:M o N:M:P.....	5
Normalización.....	5
Diseño en MySQL Workbench.....	6
Consultas.....	9
Microsoft Access.....	14
Importación de los datos.....	14
Subformularios.....	17
MongoDB.....	20
Paso de tablas a NoSQL.....	20
Banco.....	21
Pais.....	21
Cuenta.....	21
Cliente.....	22
Empleado.....	23
Consultas.....	23
Página web.....	34
HTML.....	35
CSS.....	36
JavaScript mouseover.....	39
Párrafo.....	39
Imágenes.....	39
JS sugerencias.....	41
Posible mejora.....	43
Consultas.....	43
Consulta_nombres.....	43
Consulta_transacciones.....	44
Insertar, eliminar datos.....	44
Problema surgido de array de arrays.....	44
Consulta SQL complicada.....	46

Bases de Datos

Enunciado

BankingConsulting quiere una base de datos en la que almacena la información relativa a bancos.

- Los bancos operan en países.
- Los países usan una única divisa.

- Una divisa puede ser compartida por varios países, por ejemplo el euro.
- Los clientes pueden tener varias cuentas bancarias y las cuentas bancarias pueden ser compartidas por varios clientes.
- Una cuenta está en un único banco.
- Se realizan transacciones en una divisa determinada
- Dichas transacciones tienen una única cuenta asociada
 - Cuando dos clientes se mandan dinero se generan dos transacciones, una de recibir dinero (cantidad de dinero positiva) y otra de mandar dinero (cantidad de dinero negativa)
- Los clientes piden préstamos a los bancos.
 - Un préstamo es pedido por una persona a un banco determinado.
- Los bancos tienen empleados. Estos empleados se dividen en supervisores y miembros de equipo. Un supervisor tiene varios miembros de equipo, pero un miembro de equipo es supervisado por un único supervisor.

De un país se desea guardar su Nombre y Población y un ID.

De los bancos su sede, nombre e ID.

De las divisas la tasa de cambio e ID.

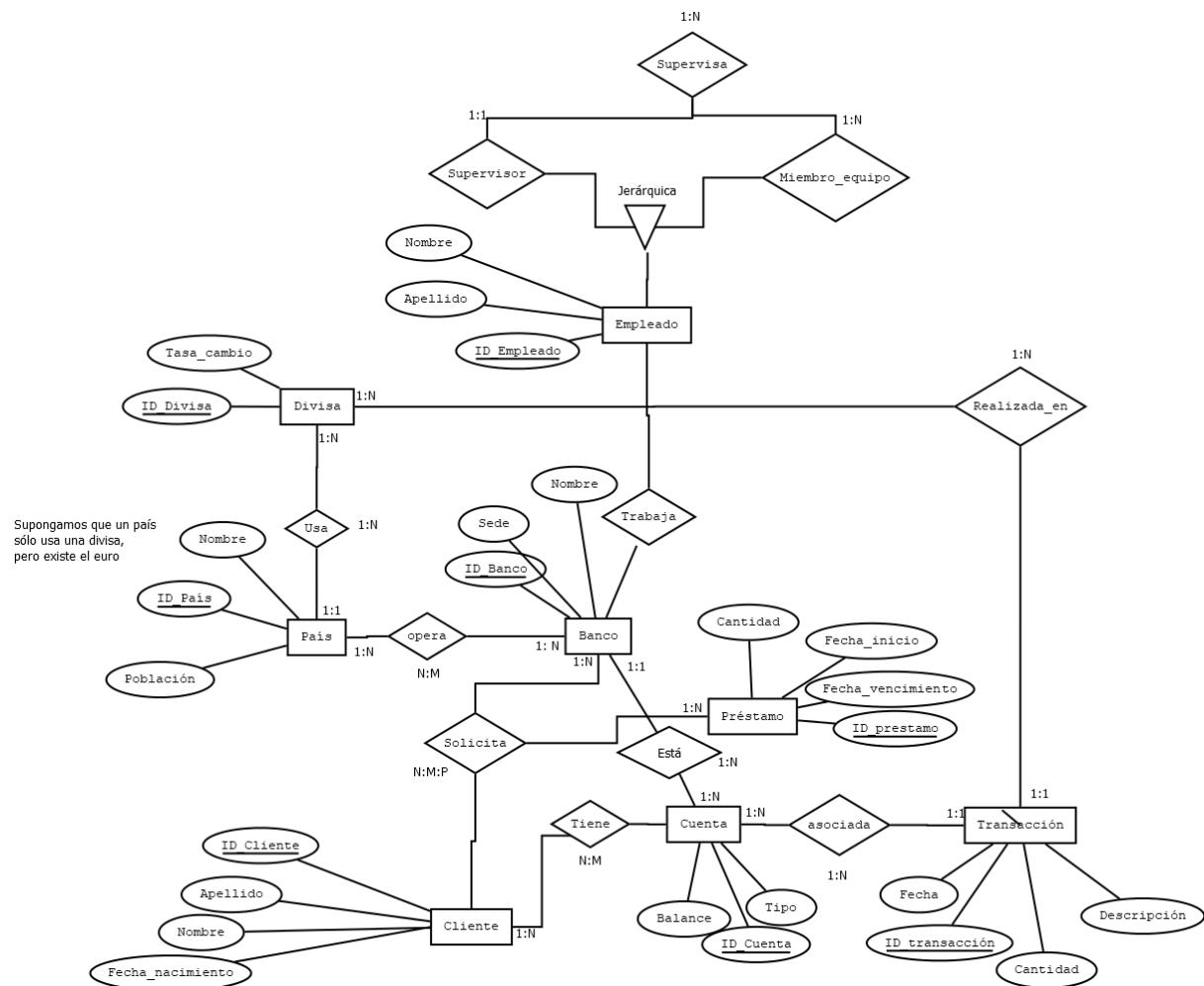
De las cuentas sus balances, tipo (depósito o cuenta corriente) e ID.

De las transacciones la fecha, cantidad, descripción e ID.

De los préstamos la cantidad, fecha inicio, fecha vencimiento e ID.

De los empleados su nombre, apellido e ID.

Diseño conceptual



Diseño lógico

Entidades

Subrayado indica que dicho atributo(s) es(son) "Primary Key"

1. País (ID_País, Nombre, Población, ID_Divisa)
 - a. FK: ID_Divisa -> Divisa
2. Banco (ID_Banco, Nombre, Sede)
3. Préstamo (ID_Préstamo, Monto, Fecha_inicio, Fecha_vencimiento)
4. Cliente (ID_Cliente, Nombre, Apellido, Fecha_nacimiento)
5. Cuenta (ID_Cuenta, Tipo, Balance, ID_Banco)
 - a. FK: ID_Banco -> Banco
6. Transacción (ID_Transacción, Fecha, Monto, Descripción, ID_Divisa, ID_Cuenta)
 - a. FK: ID_Divisa -> Divisa
 - b. FK: ID_Cuenta

7. Divisa (ID_Divisa, Código, Tasa_cambio)
8. Empleado (ID_Empleado, Nombre, Apellido, ID_Banco)
9. Supervisor(ID_Supervisor, ID_Empleado)
 - a. FK: ID_Empleado -> Empleado
10. Miembro_Equipo(ID_Miembro_Equipo, ID_Empleado, ID_Supervisor)
 - a. FK: ID_Empleado -> Empleado
 - b. FK: ID_Supervisor -> Supervisor

En Divisa el código podría haber sido “Primary Key” y en Supervisor y Miembro_Equipo el ID_Empleado podría haber funcionado como “Primary Key”, sin embargo, prefiero añadir nuevos IDs por guardar la consistencia con el resto de entidades.

Relaciones N:M o N:M:P

1. opera(ID_País, ID_Banco)
2. solicita (ID_Banco,ID_Prestamo,ID_Cliente)
3. tiene(ID_Cliente,ID_Cuenta)

Normalización

Vamos a estudiar si el esquema lógico está en 3FN.

Primero, usando la definición de clase de 1FN:

“Una tabla relacional R está en primera forma normal (1FN) si NO contiene campos multivaluados (cada atributo 1 valor).”

Como nuestro diseño no tiene campo multivaluados, concluimos que está en 1FN.

Para ver si está en 2FN es necesario que esté en 1FN y, además, que no exista dependencia transitiva entre campos no principales y claves candidatas.

- País (ID_País, Nombre, Población,ID_Divisa)
 - No hay atributos dependiente ID_País (única clave candidata)
- Banco (ID_Banco, Nombre, Sede)
 - No hay atributos parciales dependientes de la clave (ID_Banco).
- Préstamo (ID_Préstamo, Monto, Fecha_inicio, Fecha_vencimiento)
 - No hay atributos parciales dependientes de la clave (ID_Préstamo)
- Cliente (ID_Cliente, Nombre, Apellido, Fecha_nacimiento)
 - No hay atributos parciales dependientes de la clave (ID_Cliente)
- Cuenta (ID_Cuenta, Tipo, Balance, ID_Banco)
 - No hay atributos parciales dependientes de la clave (ID_Cuenta)

- Transacción (ID_Transacción, Fecha, Monto, Descripción, ID_Divisa, ID_Cuenta)
 - No hay atributos parciales dependientes de la clave (ID_Transacción)
- Divisa (ID_Divisa, Código, Tasa_cambio)
 - No hay atributos parciales dependientes de las claves candidatas (ID_Divisa y Código)
- Empleado (ID_Empleado, Nombre, Apellido, ID_Banco)
 - no hay atributos parciales dependientes de la clave (ID_Empleado)
- Supervisor(ID_Supervisor, ID_Empleado)
 - no hay atributos parciales dependientes de claves candidatas (ID_Supervisor e ID_Empleado)
- Miembro_Equipo(ID_Miembro_Equipo, ID_Empleado, ID_Supervisor)
 - no hay atributos parciales dependientes de claves candidatas (ID_Miembro_Equipo e ID_Empleado)

Luego todas las tablas están en 2FN

Y no hay dependencias transitivas, luego, como es 2FN, también es 3FN.

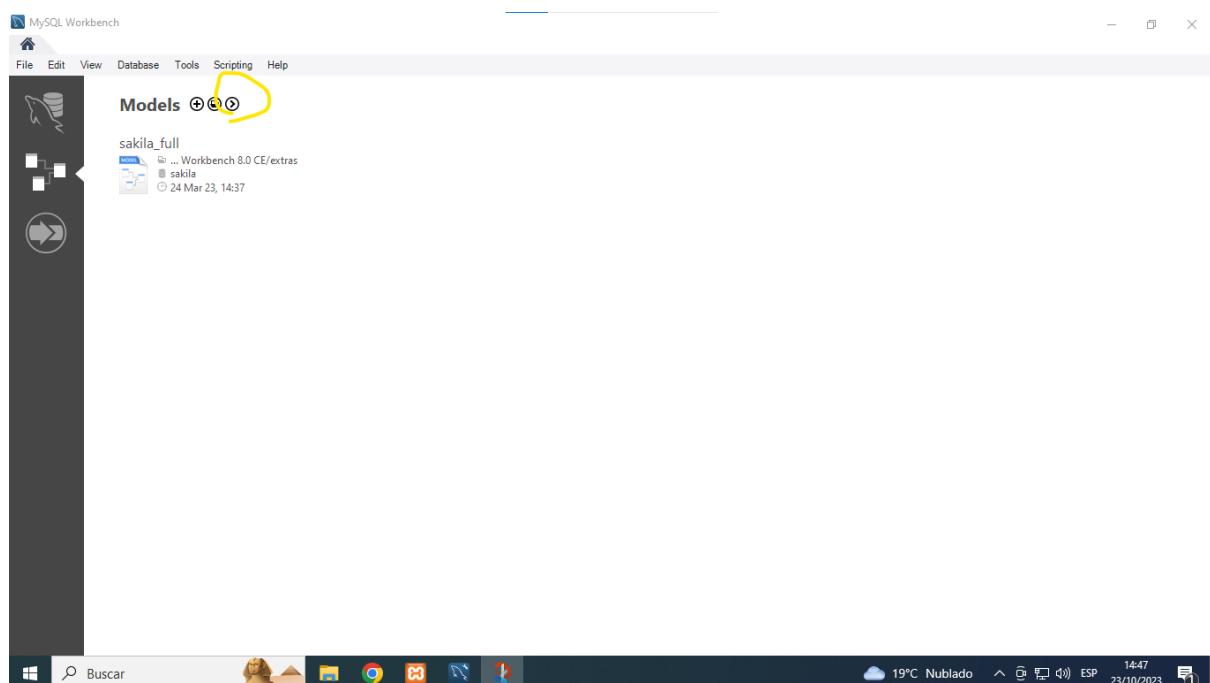
Diseño en MySQL Workbench

Primero he generado los archivos SQL:

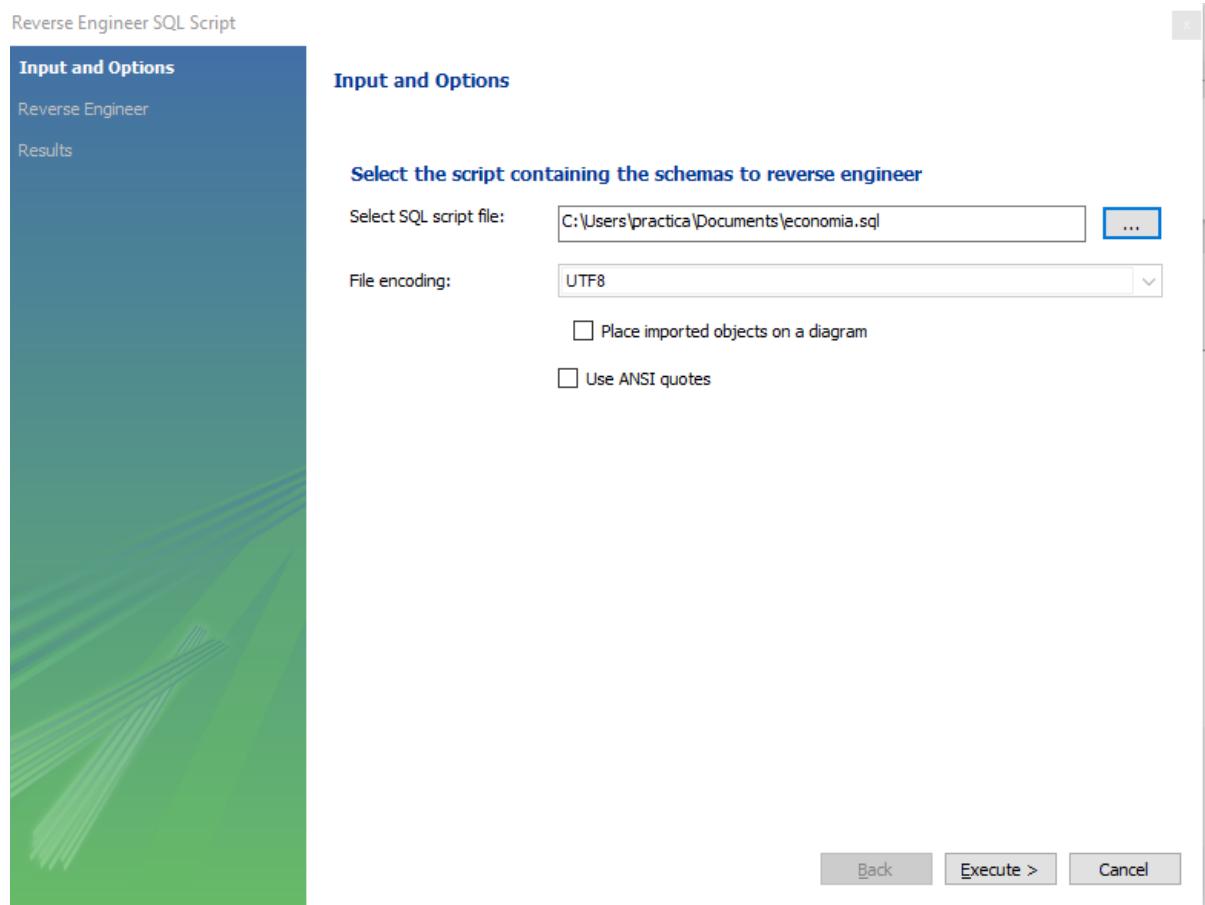
- [economiaDB.sql](#) (diseño de tablas)
- [economia_insert.sql](#) (inserción de datos)

He usado la herramienta de “reverse engineering” para hacer el diseño lógico en MySQL Workbench.

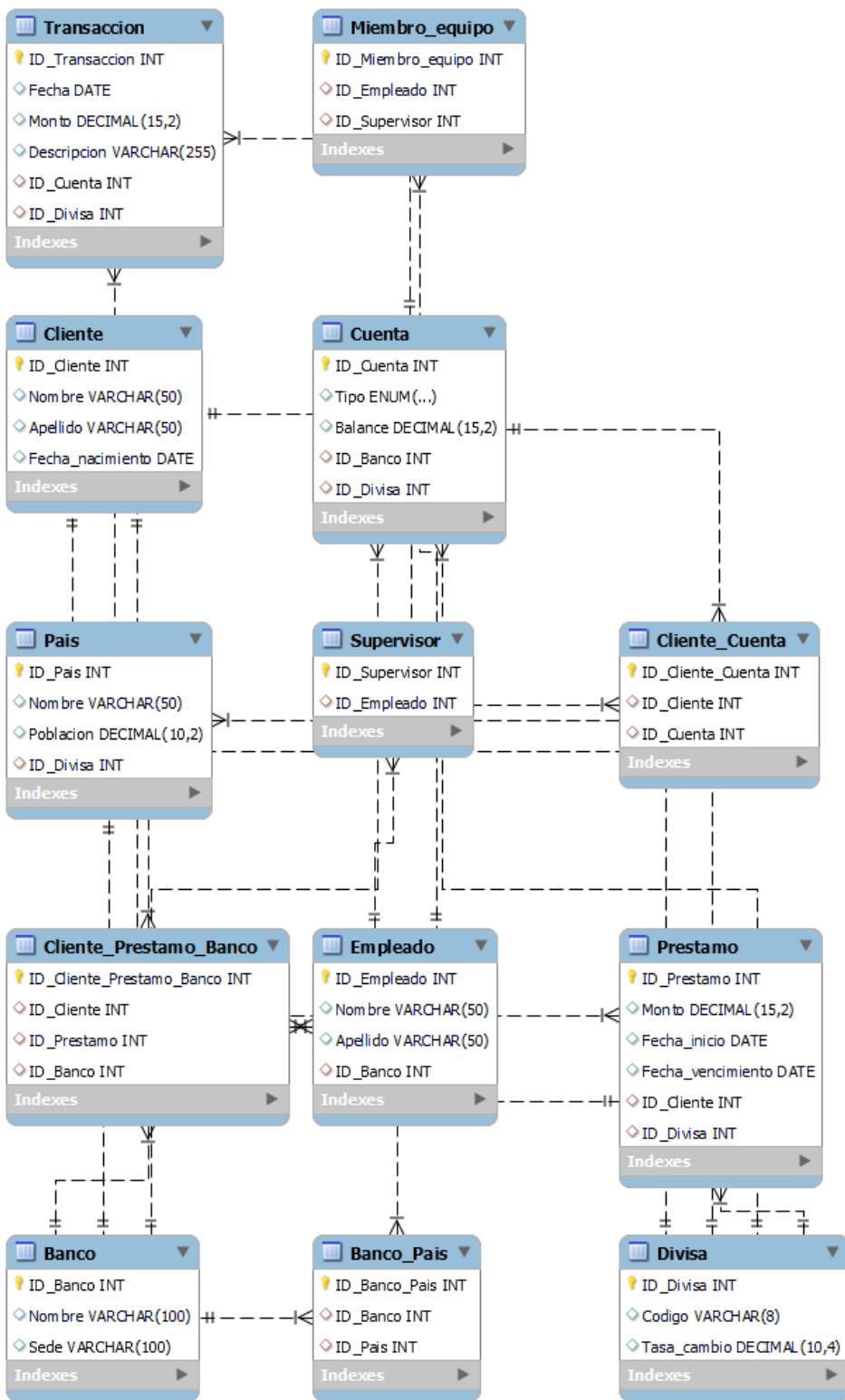
1. Hacemos click en “Create EER Model from Script”:



2. Añadimos el Script SQL donde se crean las tablas, seleccionamos “Place imported objects on a diagram” y le damos a “Execute”, luego “Next” y luego “Finish”.



Obtenemos el diagrama lógico



Archivo del modelo de MySQL workbench:

- [EconomiaDB.mwb](#)

Consultas

Las consultas están en el archivo [queries.sql](#)

1. clientes que hayan nacido después de 1960

```
SELECT * FROM Cliente WHERE Fecha_Nacimiento > '1960-12-31';
```

Result Grid			
ID_Cliente	Nombre	Apellido	Fecha_nacimiento
1	Juan	Pérez	1985-01-15
2	Ana	García	1990-06-22
3	Antonio	García	1970-01-01
4	Maximiliam	Müller	1990-01-01
5	Alexander	Zelensky	1986-01-01
11	Alex	elcapo	1987-10-06
12	Joanne	Rowling	1965-07-31
NULL	NULL	NULL	NULL

2. clientes cuyos nombres empiecen por D

```
SELECT * FROM Cliente WHERE Nombre LIKE 'D%';
```

ID_Cliente	Nombre	Apellido	Fecha_nacimiento
7	Donald	Trump	1946-01-01
9	Dalia	Lama	1938-01-01
NULL	NULL	NULL	NULL

3. clientes que tengan más de 2 cuentas

```
SELECT Nombre
FROM Cliente WHERE ID_Cliente IN
(SELECT ID_Cliente FROM Cliente_Cuenta GROUP BY ID_Cliente
 HAVING COUNT(ID_Cuenta) > 1);
```

Nombre
Juan
Ana
Antonio
Dalia

4. ids de los clientes que tengan cuenta en el Banco Bilbao y Deutsche Bank

```
SELECT ID_Cliente  
FROM Cliente_Banco WHERE ID_Banco IN  
(SELECT ID_Banco FROM Banco WHERE Nombre  
IN ('Banco Bilbao', 'Deutsche Bank'));
```

ID_Cliente
2
7

5. id de los clientes cuya posicion neta (suma de balances de sus cuentas) sea negativa

```
SELECT ID_Cliente, Nombre  
FROM Cliente WHERE ID_Cliente IN  
(SELECT ID_Cliente FROM Cuenta  
INNER JOIN Cliente_Cuenta ON Cuenta.ID_Cuenta = Cliente_Cuenta.ID_Cuenta  
GROUP BY ID_Cliente HAVING SUM(Balance) < 0);
```

ID_Cliente	Nombre
8	Jorge Mario
NULL	NULL

6. ids de las transacciones que se hagan a una cuenta con divisa distinta a la propia de la transacción

```
SELECT ID_Transaccion FROM Transaccion  
INNER JOIN Cuenta ON Transaccion.ID_Cuenta = Cuenta.ID_Cuenta  
WHERE Transaccion.ID_Divisa != Cuenta.ID_Divisa;
```

ID_Transaccion

13

7. cantidad a ingresar en la divisa de la cuenta de las transacciones que tienen una divisa distinta a la de la cuenta

```
SELECT Transaccion.Monto * D1.Tasa_cambio*1/D2.Tasa_cambio  
FROM Transaccion  
-- Join de la tabla de transacciones con la tabla de cuentas  
INNER JOIN Cuenta ON Transaccion.ID_Cuenta = Cuenta.ID_Cuenta  
-- Hago el JOIN para obtener Tasa de cambio de la moneda de la transacción  
INNER JOIN Divisa as D1 ON Transaccion.ID_Divisa = D1.ID_Divisa  
-- Hago el JOIN para obtener Tasa de cambio de la moneda de la cuenta  
INNER JOIN Divisa as D2 ON Cuenta.ID_Divisa = D2.ID_Divisa  
WHERE Transaccion.ID_Divisa != Cuenta.ID_Divisa;
```

Transaccion.Monto * D1.Tasa_cambio*1/D2.T...

191.3043478260

8. Mostar el nombre de los clientes que tenga alguna cuenta con divisa distinta a euros

```
SELECT Nombre FROM Cliente WHERE ID_Cliente IN  
(SELECT ID_Cliente FROM Cliente_Cuenta WHERE ID_Cuenta IN  
(SELECT ID_Cuenta FROM Cuenta WHERE ID_Divisa != 1));
```

Nombre

Juan

Jorge Mario

Dalia

9. Mostrar los paises que no tengan mas de 2 bancos

```
SELECT ID_Pais FROM Banco_Pais
```

```
GROUP BY ID_Pais HAVING COUNT(ID_Banco) <= 2;
```

ID_Pais
9
10

10. Mostrar las divisas que no tengan ninguna cuenta asociada

```
SELECT * FROM Divisa  
WHERE ID_Divisa NOT IN (SELECT ID_Divisa FROM Cuenta);
```

ID_Divisa	Codigo	Tasa_cambio
4	PLN	0.2200
6	NOK	0.0880
7	SEK	0.0860
NULL	NULL	NULL

11. Mostrar las divisas que se usen en menos de dos paises

```
SELECT ID_Divisa FROM Pais  
GROUP BY ID_Divisa HAVING COUNT(ID_Pais) < 2;
```

AÑADIR FOTO

12. Mostrar los prestamos que tengan una duracion mayor a 10 años

```
SELECT * FROM Prestamo WHERE DATEDIFF(Fecha_fin, Fecha_inicio) > 365*10;
```

ID_Prestamo	Monto	Fecha_inicio	Fecha_vencimien...	ID_Cliente	ID_Divisa
1	1000	2023-01-01	2023-06-30	1	4
2	2000	2023-02-01	2023-07-31	2	6
3	3000	2023-03-01	2023-08-31	3	7
4	4000	2023-04-01	2023-09-30	4	8

13. Mostrar los clientes que tengan un prestamo con una duracion mayor a 10 años

```
SELECT ID_Cliente FROM Prestamo  
WHERE DATEDIFF(Fecha_fin, Fecha_inicio) > 365*10;
```

ID_Cliente

14. Mostrar los clientes que tengan un prestamo en un banco en el que no tienen ninguna cuenta

```
SELECT Cliente_Prestamo_Banco.ID_Cliente FROM Cliente_Prestamo_Banco  
INNER JOIN Cliente_Banco  
ON Cliente_Prestamo_Banco.ID_Cliente = Cliente_Banco.ID_Cliente  
WHERE Cliente_Prestamo_Banco.ID_Banco != Cliente_Banco.ID_Banco;
```

ID_Cliente

1

15. Mostrar las divisas con las que hay una cantidad transferida total equivalente a más de 500 euros

```
SELECT Transaccion.Monto*Divisa.Tasa_cambio, Transaccion.*  
FROM Transaccion  
INNER JOIN Divisa ON Transaccion.ID_Divisa = Divisa.ID_Divisa;
```

SUM(Transaccion.Monto*Divisa.Tasa_cam... ID_Divisa

3900.000000
517.500000
832.000000

1
2
3

16. Comprobar si las cuentas asociadas que tenga el cliente está él sólo y si es así borrar la cuenta también.

Usado en [Problema surgido de array de arrays](#)

```
SELECT ID_Cuenta FROM Cliente_Cuenta  
WHERE ID_Cuenta IN  
(SELECT ID_Cuenta  
FROM Cliente_Cuenta  
GROUP BY ID_Cuenta  
HAVING COUNT(*) = 1)  
AND ID_Cliente = 1
```

Microsoft Access

Vídeos útiles:

- [Como crear Bases de Datos en Access](#)
- [Como crear Formularios en Access](#)
- [FORMULARIOS con SUBFORMULARIOS Base de Datos en ACCESS \(Parte 1\)](#)
- [» Como CREAR UN FORMULARIO en Access con BOTONES](#) 03

Importación de los datos

Exportamos los datos de phpmyadmin seleccionando la tabla que queramos exportar y los datos de la tabla que deseemos y haciendo click sobre export. Además escogemos el formato “CSV for MS Excel”.

		ID_Cliente	Nombre	Apellido	Fecha_nacimiento
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	1	Juan	Pérez	1985-01-15
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	2	Ana	García	1990-06-22
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	3	Antonio	García	1970-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	4	Maximiliam	Müller	1990-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	5	Alexander	Zelensky	1986-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	6	Marilyn	Monroe	1926-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	7	Donald	Trump	1946-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	8	Jorge Mario	Bergoglio	1936-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	9	Dalia	Lama	1938-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	10	Vladimir	Putin	1952-01-01
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	11	Alex	elcapo	1987-10-06
<input checked="" type="checkbox"/>	 Edit  Copy  Delete	12	Joanne	Rowling	1965-07-31



Check all

With selected:

 Edit

 Copy

 Delete

 Export

Luego en la foto pinchamos en Archivo de texto

En examinar escogemos el archivo csv que hemos exportado y en la imagen siguiente vemos dónde escogemos la tabla donde queremos importar los datos pulsamos aceptar y acabamos.

con los datos importados. Los cambios realizados en los datos de origen no se reflejarán en la base de datos.

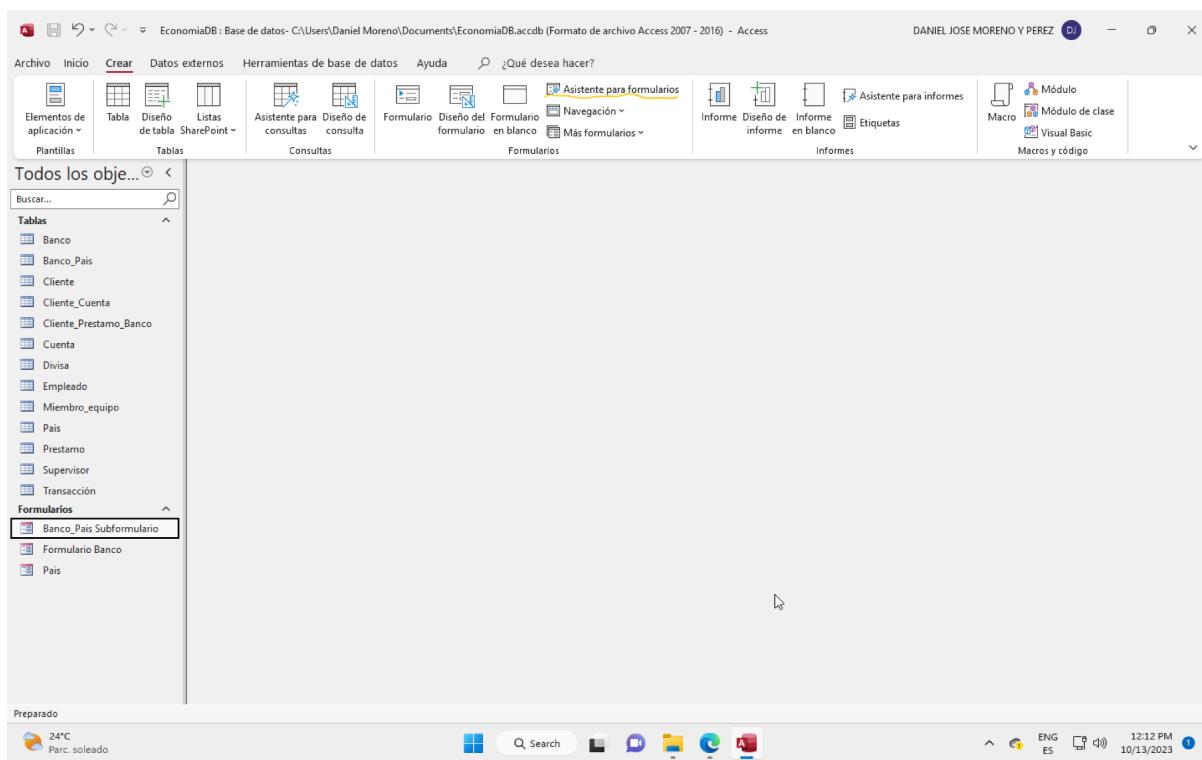
Anexar una copia de los registros a la tabla: Banco

Si la tabla especificada ya existe, Access agregará los registros a la tabla. Si la tabla no existe, Access la creará. Los cambios realizados en los datos de origen no se reflejarán en la base de datos.

Subformularios

Son útiles para mostrar datos de tablas con relaciones N:M. Por ejemplo, vamos a crear un formulario para mostrar los bancos que operan en un determinado país.

Primero pinchamos en *Asistente para formularios*.



Luego elegimos los campos que queremos que se muestren de cada tabla.

- Banco: Nombre y sede
- Banco_Pais: ID_Banco, ID_Pais
- País: Nombre

Asistente para formularios

¿Qué campos desea incluir en el formulario?
Puede elegir en más de una consulta o tabla.

Tablas/Consultas

Tabla: Banco

Campos disponibles:	Campos seleccionados:
ID_Banco	Nombre Sede

> >> < <<

Cancelar < Atrás Siguiente > Finalizar

Asistente para formularios

¿Qué campos desea incluir en el formulario?
Puede elegir en más de una consulta o tabla.

Tablas/Consultas

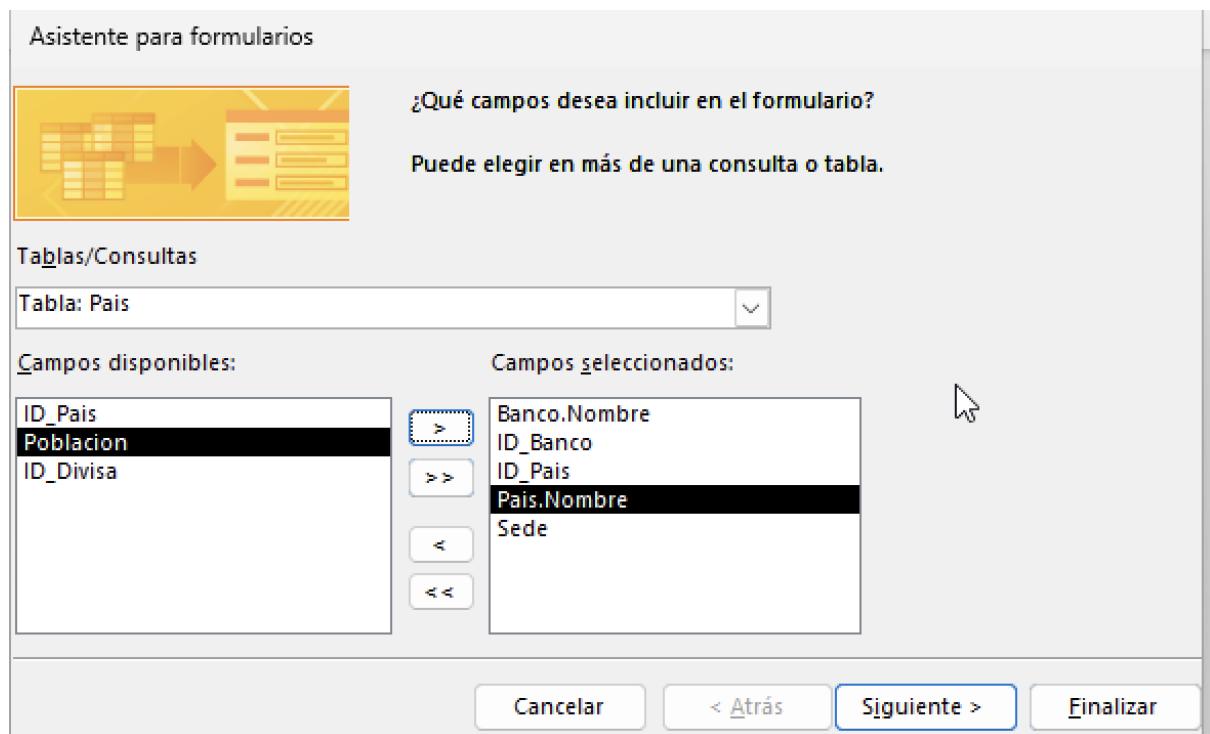
Tabla: Banco_Pais

Campos disponibles:	Campos seleccionados:
ID_Banco_Pais	Nombre ID_Banco ID_Pais Sede

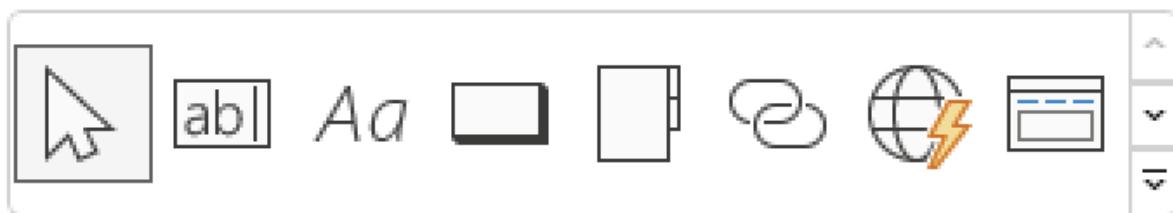
> >> < <<

Cancelar < Atrás Siguiente > Finalizar

Por último, elegimos agrupar los bancos por país.



Pinchamos en ver para ir a *vista de diseño*. Pinchamos sobre botones. Elegimos un botón para navegar a anterior y siguiente registro.



Nos queda el formulario con subformulario tal que así:

Pais2

Nombre: Reino Unido

Banco_Pais

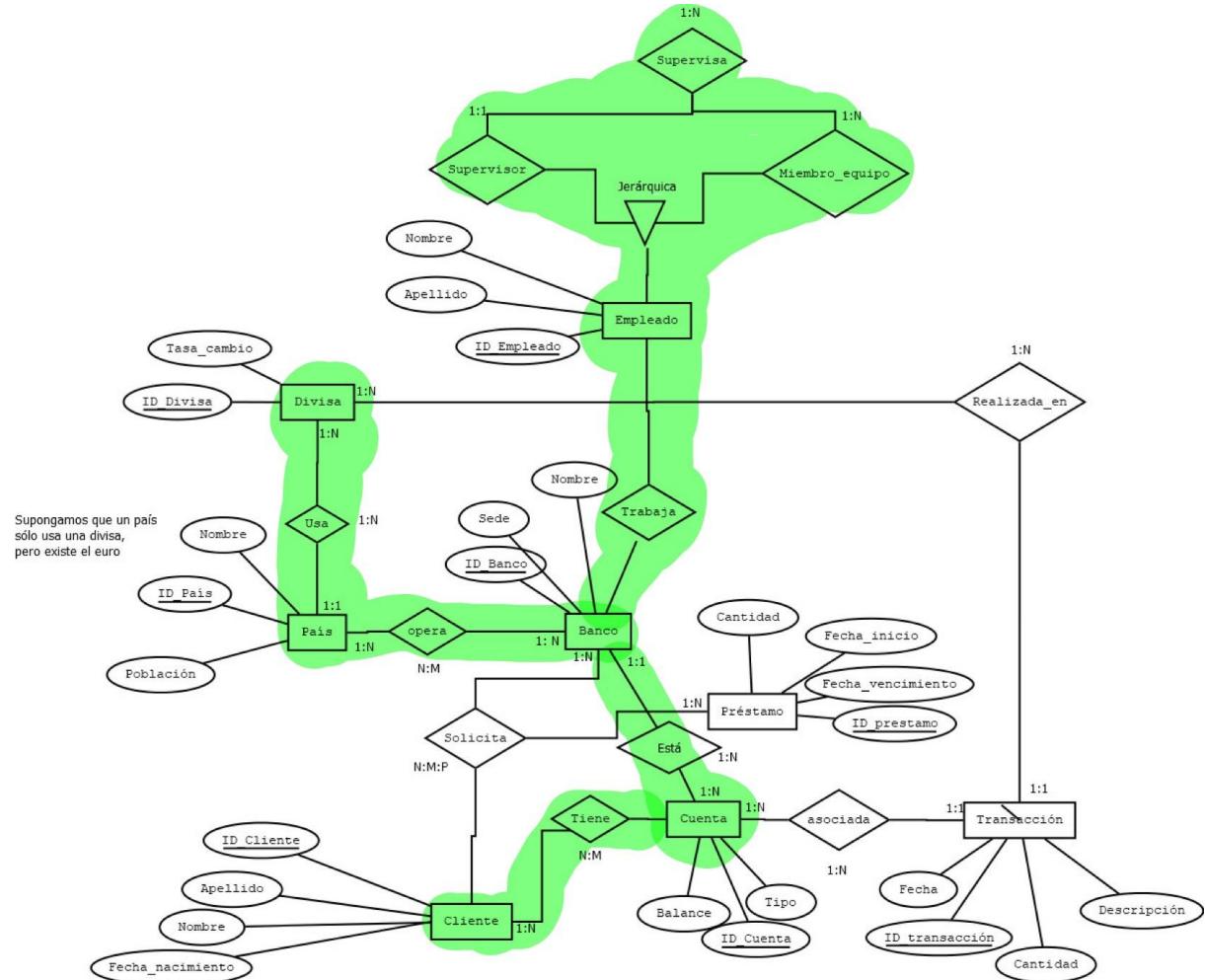
Nombre	ID_Banco
BNP Paribas	3
Crédit Agricole	4

Registro: 1 de 2 Sin filtro Buscar

MongoDB

Primero hay que adaptar la parte del modelo entidad relación a colecciones (NoSQL). En la siguiente imagen se puede ver en verde la parte del modelo que voy a adaptar.

Paso de tablas a NoSQL



Banco

Colección con Nombre, Sede, ID (los genera automáticamente Mongo DB) y para conservar la relación entre País y Banco creo una colección con un array con los IDs de País.

```
{  
  "Nombre": "Banco Santander",  
  "Sede": "Santander",  
  "Pais": [  
    1,3,6,7,8,11  
  ]  
},
```

País

Nombre, Población y colección llamada Divisa con Código y Tasa de Cambio.

```
{  
  "Nombre": "España",  
  "Poblacion": 47.0,  
  "Divisa": {  
    "Codigo": "EUR",  
    "Tasa_cambio": 1.0  
  }  
},
```

Cuenta

Divisa como subcolección y relación con Cliente mediante array de códigos.

```
{  
  "Tipo": "Ahorros",  
  "Balance": 1500.00,  
  "ID_Banco": 1,  
  "Divisa": {  
    "Codigo": "EUR",  
    "Tasa_cambio": 1.0  
  },  
  "ID_Cliente": [  
    1,2  
  ]  
},
```

Cliente

```
{  
  "ID_Cliente": 1,  
  "Nombre": "Juan",  
  "Apellido": "Pérez",  
  "Fecha_nacimiento": "1985-01-15"  
},
```

Empleado

Para adaptar la relación jerárquica vamos a poner un campo Supervisor_ID, que será null para los supervisores (no son supervisados por nadie) pero cuando sea un miembro de un equipo (no supervisor) tendrá el ID del supervisor asignado.

```
{  
    "ID_Emppleado": 1,  
    "Nombre": "Pedro",  
    "Apellido": "Martínez",  
    "ID_Banco": 1,  
    "Supervisor_ID": 4  
,
```

Los archivos JSON anteriores se encuentran en github:

- [Banco.json](#)
- [Cliente.json](#)
- [Cuenta.json](#)
- [Empleado.json](#)
- [Pais.json](#)

Consultas

Las consultas siguientes se encuentran en el siguiente fichero [consultas.txt](#)

```
(base) dankd@Dani-Moreno-senor-y-caballero-portatil ~ % mongosh  
Current Mongosh Log ID: 6530facd073216c4606a7bdf  
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.9.1  
Using MongoDB: 6.0.6  
Using Mongosh: 1.9.1  
  
For mongosh info see: https://docs.mongodb.com/mongodb-shell/  
  
-----  
The server generated these startup warnings when booting  
2023-10-19T11:01:10.530+02:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted  
-----  
  
test> show dbs  
EconomiaDB 200.00 KiB  
admin      320.00 KiB  
config     108.00 KiB  
local      16.27 MiB  
test> use EconomiaDB  
switched to db EconomiaDB
```

1. Obtener todos los bancos con sede en "Londres"

```
EconomiaDB.Banco.find({ "Sede": "Londres" })
```

```
EconomiaDB> db.Cuenta.find({ "Tipo": "Ahorros", "Balance": { $gt: 1000 } })
[
  {
    _id: ObjectId("6530fa676b8946bc00fb0643"),
    Tipo: 'Ahorros',
    Balance: 1500,
    ID_Banco: 1,
    Divisa: { Código: 'EUR', Tasa_cambio: 1 },
    ID_Cliente: [ 1, 2 ]
  },
  {
    _id: ObjectId("6530fa676b8946bc00fb0645"),
    Tipo: 'Ahorros',
    Balance: 2000,
    ID_Banco: 3,
    Divisa: { Código: 'EUR', Tasa_cambio: 1 },
    ID_Cliente: [ 5, 6 ]
  }
]
```

2. Encontrar todas las cuentas de ahorro con un saldo mayor de 1000 euros

```
db.Cuenta.find({ "Tipo": "Ahorros", "Balance": { $gt: 1000 } })
```

```
[EconomiaDB> db.Banco.find({ 'Sede': 'Londres' })
[
  {
    _id: ObjectId("6530fd336b8946bc00fb0655"),
    Nombre: 'Barclays',
    Sede: 'Londres',
    País: [ 6, 1, 7, 8, 11, 12 ]
  },
  {
    _id: ObjectId("6530fd336b8946bc00fb065a"),
    Nombre: 'Lloyds Banking Group',
    Sede: 'Londres',
    País: [ 1, 3, 7, 8, 11, 12 ]
  }
]
```

3. Buscar empleados que trabajen en un banco con sede en "Londres"

```
db.Empleado.find({ "ID_Banco": { $in: [2, 7] } })
```

```
EconomiaDB> db.Empleado.find({ "ID_Banco": { $in: [2, 7] } })
[
  {
    _id: ObjectId("65266e4ed74c8d087ed328c2"),
    ID_Emppleado: 3,
    Nombre: 'Luis',
    Apellido: 'González',
    Fecha_contratacion: '2016-09-01',
    ID_Banco: 2,
    Supervisor_ID: 4
  },
  {
    _id: ObjectId("65266e4ed74c8d087ed328c4"),
    ID_Emppleado: 5,
    Nombre: 'Ariana',
    Apellido: 'Grande',
    Fecha_contratacion: '1998-07-11',
    ID_Banco: 2,
    Supervisor_ID: 9
  },
  {
    _id: ObjectId("65266e4ed74c8d087ed328c9"),
    ID_Emppleado: 10,
    Nombre: 'Johnny',
    Apellido: 'Depp',
    Fecha_contratacion: '2023-01-19',
    ID_Banco: 7,
    Supervisor_ID: 12
  }
]
```

4. Encontrar todos los clientes cuyo nombre sea "Juan"

```
db.Cliente.find({ "Nombre": "Juan" })
```

```
EconomiaDB>      db.Cliente.find({ "Nombre": "Juan" })
[
  {
    _id: ObjectId("65266e3fd74c8d087ed328b3"),
    ID_Cliente: 1,
    Nombre: 'Juan',
    Apellido: 'Pérez',
    Fecha_nacimiento: '1985-01-15'
  }
]
```

5. Obtener la población de todos los países con una población mayor de 50 millones de habitantes

```
db.Pais.find({ "Poblacion": { $gt: 50 } })
```

```
EconomiaDB> db.Pais.find({ "Poblacion": { $gt: 50 } })
[ {
  _id: ObjectId("652500cad74c8d087ed328a6"),
  Nombre: 'Reino Unido',
  Poblacion: 67,
  Divisa: { Código: 'GBP', Tasa_cambio: 1.15 }
},
{
  _id: ObjectId("652500cad74c8d087ed328a7"),
  Nombre: 'Alemania',
  Poblacion: 83.2,
  Divisa: { Código: 'EUR', Tasa_cambio: 1 }
},
{
  _id: ObjectId("652500cad74c8d087ed328aa"),
  Nombre: 'Francia',
  Poblacion: 67.75,
  Divisa: { Código: 'EUR', Tasa_cambio: 1 }
},
{
  _id: ObjectId("652500cad74c8d087ed328af"),
  Nombre: 'Italia',
  Poblacion: 59.11,
  Divisa: { Código: 'EUR', Tasa_cambio: 1 }
}
]
```

6. Encontrar todas las cuentas en euros (EUR) con un saldo mayor de 1000 euros

```
db.Cuenta.find({ "Divisa.Código": "EUR", "Balance": { $gt: 1000 } })
```

```
[EconomiaDB> db.Cuenta.find({ "Divisa.Codigo": "EUR", "Balance": { $gt: 1000 } })
[
  {
    _id: ObjectId("6530fa676b8946bc00fb0643"),
    Tipo: 'Ahorros',
    Balance: 1500,
    ID_Banco: 1,
    Divisa: { Código: 'EUR', Tasa_cambio: 1 },
    ID_Cliente: [ 1, 2 ]
  },
  {
    _id: ObjectId("6530fa676b8946bc00fb0645"),
    Tipo: 'Ahorros',
    Balance: 2000,
    ID_Banco: 3,
    Divisa: { Código: 'EUR', Tasa_cambio: 1 },
    ID_Cliente: [ 5, 6 ]
  },
  {
    _id: ObjectId("6530fa676b8946bc00fb0646"),
    Tipo: 'Corriente',
    Balance: 50000,
    ID_Banco: 5,
    Divisa: { Código: 'EUR', Tasa_cambio: 1 },
    ID_Cliente: []
  },
  {
    _id: ObjectId("6530fa676b8946bc00fb0649"),
    Tipo: 'Corriente',
    Balance: 40000,
    ID_Banco: 4,
    Divisa: { Código: 'EUR', Tasa_cambio: 1 },
    ID_Cliente: []
  }
]
```

7. Encontrar todos los empleados que no tienen supervisor, es decir, los supervisores.

```
db.Empleado.find({ "Supervisor_ID": null })
```

```
[EconomiaDB> db.Empleado.find({ "Supervisor_ID": null })
[
  {
    _id: ObjectId("65266e4ed74c8d087ed328c3"),
    ID_Emppleado: 4,
    Nombre: 'Angelina',
    Apellido: 'Jolie',
    Fecha_contratacion: '1999-11-23',
    ID_Banco: 1,
    Supervisor_ID: null
  },
  {
    _id: ObjectId("65266e4ed74c8d087ed328c8"),
    ID_Emppleado: 9,
    Nombre: 'George',
    Apellido: 'Lucas',
    Fecha_contratacion: '1998-04-19',
    ID_Banco: 6,
    Supervisor_ID: null
  },
  {
    _id: ObjectId("65266e4ed74c8d087ed328cb"),
    ID_Emppleado: 12,
    Nombre: 'Mohamed',
    Apellido: 'bin Salmán',
    Fecha_contratacion: '2021-06-29',
    ID_Banco: 9,
    Supervisor_ID: null
  }
]
```

8. Encontrar los bancos que operan en más de 5 países

```
db.Banco.aggregate([
{
  $project: {
    Nombre: 1,
    Sede: 1,
    NumPaises: { $size: "$Pais" }
  }
},
{
  $match: {
    NumPaises: { $gt: 5 }
  }
}]
```

```

        }
    }
])

ó

db.Banco.find({
    $where: "this.Pais.length > 5"
})

[
    {
        _id: ObjectId("6530fd336b8946bc00fb0654"),
        Nombre: 'Banco Santander',
        Sede: 'Santander',
        NumPaises: 6
    },
    {
        _id: ObjectId("6530fd336b8946bc00fb0655"),
        Nombre: 'Barclays',
        Sede: 'Londres',
        NumPaises: 6
    },
    {
        _id: ObjectId("6530fd336b8946bc00fb0656"),
        Nombre: 'BNP Paribas',
        Sede: 'Paris',
        NumPaises: 8
    },
    {
        _id: ObjectId("6530fd336b8946bc00fb0657"),
        Nombre: 'Crédit Agricole',
        Sede: 'Paris',
        NumPaises: 12
    },
    {
        _id: ObjectId("6530fd336b8946bc00fb0659"),
        Nombre: 'Deutsche Bank',
        Sede: 'Frankfurt',
        NumPaises: 6
    },
    {
        _id: ObjectId("6530fd336b8946bc00fb065a"),
        Nombre: 'Lloyds Banking Group',
        Sede: 'Londres',
        NumPaises: 6
    },
    {
        _id: ObjectId("6530fd336b8946bc00fb065b"),
        Nombre: 'Intensa Sanpaolo',
        Sede: 'Turin',
        NumPaises: 8
    }
]

```

9. Obtener el nombre y apellido de los empleados que trabajan en el banco con ID 1

```
db.Empleado.find({ "ID_Banco": 1 }, { "Nombre": 1, "Apellido": 1 })
```

```
[EconomiaDB> db.Empleado.find({ "ID_Banco": 1 }, { "Nombre": 1, "Apellido": 1 })
[  {
      _id: ObjectId("65266e4ed74c8d087ed328c0"),
      Nombre: 'Pedro',
      Apellido: 'Martínez'
    },
    {
      _id: ObjectId("65266e4ed74c8d087ed328c1"),
      Nombre: 'Carla',
      Apellido: 'Fernández'
    },
    {
      _id: ObjectId("65266e4ed74c8d087ed328c3"),
      Nombre: 'Angelina',
      Apellido: 'Jolie'
    }
]
```

10. Encontrar las cuentas de ahorro con un saldo negativo

```
db.Cuenta.find({ "Tipo": "Ahorros", "Balance": { $lt: 0 } })
```

CORREGIR

11. Encontrar las cuentas de ahorro con un saldo negativo

```
db.Cuenta.find({ "Tipo": "Ahorros", "Balance": { $lt: 0 } })
```

12. Encuentra los tres bancos con el mayor número de sucursales (países en los que operan)

```
db.Banco.aggregate([
  { $project: { "Nombre": 1, "Numero_Paises": { $size: "$Pais" } } },
  { $sort: { "Numero_Paises": -1 } },
  { $limit: 3 }
])
```

```
EconomiaDB> db.Banco.aggregate([
...   { $project: { "Nombre": 1, "Numero_Paises": { $size: "$Pais" } } },
...   { $sort: { "Numero_Paises": -1 } },
...   { $limit: 3 }
... ])
[  {
      _id: ObjectId("6530fd336b8946bc00fb0657"),
      Nombre: 'Crédit Agricole',
      Numero_Paises: 12
    },
    {
      _id: ObjectId("6530fd336b8946bc00fb065b"),
      Nombre: 'Intensa Sanpaolo',
      Numero_Paises: 8
    },
    {
      _id: ObjectId("6530fd336b8946bc00fb0656"),
      Nombre: 'BNP Paribas',
      Numero_Paises: 8
    }
]
```

13. Encuentra el cliente más joven

```
db.Cliente.aggregate([
```

```

{
  $project: {
    "Nombre": 1,
    "Apellido": 1,
    "Fecha_nacimiento": { $toDate: "$Fecha_nacimiento" },
    "Edad": {
      $floor: {
        $divide: [
          { $subtract: [new Date(), { $toDate: "$Fecha_nacimiento" }] },
          31536000000 // Milisegundos en un año
        ]
      }
    }
  },
  { $sort: { "Edad": 1 } },
  { $limit: 1 }
}
])

EconomiaDB> db.Cliente.aggregate([
...   {
...     $project: {
...       "Nombre": 1,
...       "Apellido": 1,
...       "Fecha_nacimiento": { $toDate: "$Fecha_nacimiento" },
...       "Edad": {
...         $floor: {
...           $divide: [
...             { $subtract: [new Date(), { $toDate: "$Fecha_nacimiento" }] },
...             31536000000 // Milisegundos en un año
...           ]
...         }
...       }
...     },
...     { $sort: { "Edad": 1 } },
...     { $limit: 1 }
...   }
]
[ {
  _id: ObjectId("65266e3fd74c8d087ed328b4"),
  Nombre: 'Ana',
  Apellido: 'García',
  Fecha_nacimiento: ISODate("1990-06-22T00:00:00.000Z"),
  Edad: 33
}
]

```

14. Encuentra el saldo total de todas las cuentas en euros (EUR) para cada banco

```

db.Cuenta.aggregate([
  { $match: { "Divisa.Codigo": "EUR" } },
  {
    $group: {
      _id: "$ID_Banco",

```

```

        Saldo_Total: { $sum: "$Balance" }
    }
}
])
EconomiaDB> db.Cuenta.aggregate([
...   { $match: { "Divisa.Codigo": "EUR" } },
...   {
...     $group: {
...       _id: "$ID_Banco",
...       Saldo_Total: { $sum: "$Balance" }
...     }
...   }
... ])
[
  { _id: 6, Saldo_Total: 200 },
  { _id: 5, Saldo_Total: 50000 },
  { _id: 3, Saldo_Total: 2000 },
  { _id: 1, Saldo_Total: 1500 },
  { _id: 4, Saldo_Total: 40000 },
  { _id: 10, Saldo_Total: 900 },
  { _id: 9, Saldo_Total: 200 }
]

```

15. Encuentra la edad promedio de los empleados en el banco con ID 1

```

db.Empleado.aggregate([
{
  $match: { "ID_Banco": 1 }
},
{
  $project: {
    "Fecha_contratacion": { $toDate: "$Fecha_contratacion" },
    "Edad": {
      $floor: {
        $divide: [
          { $subtract: [new Date(), { $toDate: "$Fecha_contratacion" }] },
          31536000000 // Milisegundos en un año
        ]
      }
    }
  }
}
]

```

```

},
{
  $group: {
    _id: null,
    EdadPromedio: { $avg: "$Edad" }
  }
}
])

EconomiaDB> db.Empleado.aggregate([
...   {
...     $match: { "ID_Banco": 1 }
...   },
...   {
...     $project: {
...       "Fecha_contratacion": { $toDate: "$Fecha_contratacion" },
...       "Edad": {
...         $floor: {
...           $divide: [
...             { $subtract: [new Date(), { $toDate: "$Fecha_contratacion" }] },
...             31536000000 // Milisegundos en un año
...           ]
...         }
...       }
...     },
...   },
...   {
...     $group: {
...       _id: null,
...       EdadPromedio: { $avg: "$Edad" }
...     }
...   }
... ])
[ { _id: null, EdadPromedio: 23 } ]

```

16. Encuentra los clientes que no tienen cuentas en ningún banco.

```

db.Cliente.find({
  "ID_Cliente": { $nin: db.Cuenta.distinct("ID_Cliente") }
})

```

```

EconomiaDB> db.Cliente.find({
...   "ID_Cliente": { $nin: db.Cuenta.distinct("ID_Cliente") }
... })
[
  {
    _id: ObjectId("65266e3fd74c8d087ed328bb"),
    ID_Cliente: 9,
    Nombre: 'Dalia',
    Apellido: 'Lama',
    Fecha_nacimiento: '1938-01-01'
  },
  {
    _id: ObjectId("65266e3fd74c8d087ed328bd"),
    ID_Cliente: 11,
    Nombre: 'Alex',
    Apellido: 'elcapo',
    Fecha_nacimiento: '1987-10-06'
  },
  {
    _id: ObjectId("65266e3fd74c8d087ed328be"),
    ID_Cliente: 12,
    Nombre: 'Joanne',
    Apellido: 'Rowling',
    Fecha_nacimiento: '1965-07-31'
  }
]

```

Página web

Bankconsulting nos ha pedido diseñar una página web en la que puedan describir la empresa y su filosofía. Además, quieren poder filtrar los nombres de los clientes por su nombre y/o apellidos y ver las consultas más nuevas de una fecha dada y una divisa seleccionada.

Se puede visitar la web en github: <https://danmorper.github.io/> pero no tiene la parte de servidor.

Primero creamos el archivo [index.html](#). Luce tal que así:

The screenshot shows a website layout for 'BankConsulting'. At the top, there's a dark header bar with the 'BankConsulting' logo on the left and three navigation links: 'Sobre nosotros', 'Consulta nombres', and 'Consulta transacciones'. Below the header is a main content area with a light blue background. The title 'BankConsulting' is centered at the top of this area, followed by the tagline 'Tu socio en el mundo de la banca comercial.' Below the tagline is a large image of a hand holding a lit lightbulb, with dashed arrows forming a circular path around it, symbolizing ideas and innovation. A small caption below the image reads: 'Somos una consultoría especializada en servicios para bancos comerciales, brindando soluciones innovadoras y estratégicas para impulsar tu éxito.' Underneath this section is a dark horizontal bar containing the title 'Nuestro Enfoque de Trabajo en BankConsulting'. Below this bar is another light blue section with the same title. It contains a paragraph about their collaborative approach and two numbered points under the heading 'Trabajo en Equipo y Comunicación Fluida'. The first point discusses how their team works together and communicates fluidly. The second point is titled 'Reputados Profesionales' and highlights their experienced professionals. At the bottom of the page is a contact email: 'Contacto: info@bankconsulting.com'.

He usado bootstrap5 para estructurar la página.

Se compone de una barra de navegación fija en la parte de arriba de la pantalla y de una parte inferior que cambia según donde pulses en la barra de navegación.

Si pinchas en “sobre nosotros” ves una breve descripción de lo que es BankConsulting (por defecto cuando abres la web).

Si pinchas sobre “Consulta nombres” te aparece un formulario en el que escribes nombre y/o apellidos de el/los cliente/s que quieras filtrar. Al pinchar sobre “Enviar” te aparecen los clientes que cuadran con la búsqueda en la tabla inferior.

Si pinchas sobre “Consulta transacciones” te aparecen 3 columnas.

- En la de la izquierda introduces la fecha a partir de la cual quieras ver transacciones. Por ejemplo, si introduces el 1 de Enero de 2020, te saldrán transacciones más recientes de dicha fecha, es decir, una transacción que se hizo el 4 de Abril de 2021 podría aparecer.
- En la columna central aparecen diferentes divisas. Sólo aparecerán las transacciones realizadas en las divisas que estén seleccionadas.
- En la columna de la derecha aparecen dos botones.
 - Enviar: hacer la búsqueda de las transacciones
 - Reset: restablecer todos los valores al predeterminado.

HTML

Barra de Navegación de bootstrap5:

```
<div class="navbar navbar-expand bg-secondary navbar-dark fixed-top">
  <nav class="navbar navbar-expand bg-secondary navbar-dark">

    <a class="navbar-brand" href="#s1" data-bs-toggle="tab">
      
    </a>

    <ul class="navbar-nav nav">
      <li class="nav-item">
        <a href="#s1" class="nav-link active" data-bs-toggle="tab">Sobre nosotros</a>
      </li>
      <li class="nav-item">
        <a href="#s2" class="nav-link" data-bs-toggle="tab">Consulta nombres</a>
      </li>
      <li class="nav-item">
        <a href="#s3" class="nav-link" data-bs-toggle="tab">Consulta transacciones</a>
      </li>
```

Contenido de las diferentes secciones de la barra de navegación:

```
56 <div class="tab-content">
57   <!-- Quiénes somos -->
58   <div class="tab-pane active abajo" id="s1">
59     <div class="flex-container">
60       <div class="caja">
61         <h1>BankConsulting</h1>
62         <p class="cambiar">Tu socio en el mundo de la banca comercial.</p>
63         Somos una consultoría especializada en servi
65       </div>
66       <div class="caja">
67         <h1>Nuestro Enfoque de Trabajo en BankConsulting</h1>
68
69         <p class="cambiar">En BankConsulting, creemos firmemente que el é
70
71         <ol>
72           <li><h2>Trabajo en Equipo y Comunicación Fluida</h2></li>
73
74             <p class="cambiar">Para nosotros, el trabajo en equipo es más q
75
76             La comunicación fluida es el pegamento que mantiene unido nuest
77
78           <li><h2>Reputados Profesionales</h2></li>
79
80             <p class="cambiar">En BankConsulting, nos enorgullecemos de con
```

La web tiene un footer con el contacto

```
201
202      <footer>
203          <p class="cambiar bg-info">Contacto: info@bankconsulting.com</p>
204      </footer>
```

En el head del html hemos añadido la hoja de estilos de bootstrap y una creada por mí, [style.css](#)

```
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6
7   <!-- Añadimos hoja de estilos Bootstrap -->
8   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
9   <link rel="stylesheet" href="style_main.css">
10  <title>BankConsulting – Consultoría de Bancos Comerciales</title>
11 </head>
```

css

- flex-container:
 - **flex-direction column**, hace que los div siguientes (en este caso todos tienen clase “caja”) aparezcan uno debajo de otro

CSS Demo: flex-direction

RESET

flex-direction: row;

flex-direction: row-reverse;

flex-direction: column;

flex-direction: column-reverse;

The diagram illustrates the visual outcome of four different flex-direction settings. On the left, four boxes contain the CSS code for each direction. On the right, the resulting layout is shown in a grid:

- flex-direction: row;**: Three items arranged horizontally from left to right.
- flex-direction: row-reverse;**: Three items arranged horizontally from right to left.
- flex-direction: column;**: Three items arranged vertically from top to bottom.
- flex-direction: column-reverse;**: Three items arranged vertically from bottom to top.

fuente: <https://developer.mozilla.org/en-US/docs/Web/CSS/flex-direction>

- caja
- img
 - edita tamaño imágenes
- logo
 - edita el tamaño, lo creé específicamente para el logo de la barra de navegación
- cambiar
 - Te hace la letra más grande
- nuevaClase
 - se usa en js
- nuevaClase2
 - se usa en js

- abajo
 - Hace el margen superior más grande para desplazar el contenido hacia abajo y que no lo tape la barra de navegación

```
1  .flex-container {  
2      display: flex;  
3      flex-direction: column;  
4      flex-wrap: wrap;  
5      justify-content: space-evenly;  
6      background-color: □rgb(42, 59, 74);  
7  }  
8  
9  .caja {  
10     text-align: center;  
11     border-radius: 10px;  
12     background: ■#b2cfe4;  
13     margin: 10px;  
14  }  
15  
16  img {  
17     width: 600px;  
18     height: 400px;  
19  }  
20  
21  .logo {  
22     width: 100px;  
23     height: 100px;  
24  }  
25  
26  .cambiar {  
27     font-size: 20px;  
28  }
```

```
29
30     .nuevaClase{
31         font-weight: bolder;
32     }
33
34     .nuevaClase2{
35         width: 900px;
36         height: 700px;
37     }
38
39     .abajo{
40         margin-top: 140px;
41     }
```

JavaScript mouseover

Quiero que cuando pase el ratón por encima de:

1. un párrafo, la letra se ponga en negrita
2. una imagen, se haga más grande

Párrafo

Seleccionamos todos los párrafos que tienen la clase “cambiar” (los tienen todos, lo he puesto yo). Lo guardamos en variable `parrafos`.

con `forEach` hacemos dos event listener:

- `mouseover` -> cuando pasemos el ratón por encima añade la clase de css “nuevaClase”
- `mouseout` -> cuando quitemos el ratón de encima quita la clase de css “nuevaClase”

Imágenes

Sigue el mismo principio, pero con `nuevaClase2` y se seleccionan las imágenes.

```
1 // Cambiar el color de los párrafos al pasar el ratón por encima
2
3 const parrafos = document.querySelectorAll(".cambiar");
4
5 parrafos.forEach(parrafo) => {
6     parrafo.addEventListener("mouseover", () => {
7         parrafo.classList.add("nuevaClase");
8     });
9
10    parrafo.addEventListener("mouseout", () => {
11        parrafo.classList.remove("nuevaClase");
12    });
13};
14
15 // Cambiar el tamaño de las imágenes al pasar el ratón por encima
16
17 const imagenes = document.querySelectorAll(".imagen");
18
19 imagenes.forEach(imagen) => {
20     imagen.addEventListener("mouseover", () => {
21         imagen.classList.add("nuevaClase2");
22     });
23
24     imagen.addEventListener("mouseout", () => {
25         imagen.classList.remove("nuevaClase2");
26     });
27 };
28 };
```

JS sugerencias

Sobre nosotros Consulta nombres Consulta transacciones Insertar datos **Sugerencias**

Escribe tus Sugerencias

Título de la sugerencia:

Sugerencias realizadas:

Contacto: info@bankconsulting.com

Hay un input text para escribir título de la sugerencia y textarea para escribir la sugerencia. Al pinchar sobre “Finalizar” se añade lo que se haya escrito después de “Sugerencias realizadas”. El título se mostrará pero la sugerencia estará oculta.

```

29 // Añadir div cuando se haga click en el botón con id "boton-sugerencia"
30
31 const boton = document.querySelector("#boton-sugerencia");
32 let sugerencias = document.querySelector("#anadir-sugerencias");
33 let num_sugerencias = 0; // Hay 0 sugerencias
34
35 boton.addEventListener("click", () => {
36     // Tomo titulo y texto de textarea
37     let titulo = document.querySelector("#titulo").value;
38     let texto = document.querySelector("#texto-sugerencia").value;
39     console.log(texto);
40     if (titulo == "" || texto == "") {
41         alert("Debes llenar todos los campos");
42     } else {
43         num_sugerencias += 1; // Añado una sugerencia
44         console.log(num_sugerencias);
45
46         // Creo 3 div uno para el titulo, otro para el texto y otro para los dos anteriores
47         let div = document.createElement("div");
48         div.id = "divid" + num_sugerencias; // Le pongo una id para usarla luego para mostrar el texto
49         div.classList.add("sugerencia"); // Le añado la clase sugerencia para usar queryselectorall
50         div.innerHTML += "<h3 onclick='aparece(event)' id='"+titulo${num_sugerencias}>" + titulo + "</h3>";
51         div.innerHTML += "<p id='"+texto${num_sugerencias}>" + texto + "</p>";
52         sugerencias.appendChild(div);
53
54         // Limpio titulo y textarea
55         document.getElementById("titulo").value = "";
56         document.querySelector("#texto-sugerencia").value = "";
57
58         // // Ocultar texto de la sugerencia que hemos añadido
59         document.getElementById(`texto${num_sugerencias}`).style.display = "none";
60     }
61 });
62 });

```

Cuando pincho sobre botón primero veo si no se ha añadido nada como título o sugerencia, en caso afirmativo saco alerta y NO CAMBIO num_sugerencias.

Si tengo título Y sugerencia, creo un elemento div que va a tener como hijos al título, con etiqueta h3, y texto de la sugerencia, con etiqueta p (que lo oculta al final).

Posteriormente, dejo en blanco el input y el textarea.

```

65 // Funcion para mostrar el texto de la sugerencia
66 function aparece(event) {
67     let idtitulo = event.target.id; // Cojo el id del titulo
68     console.log(idtitulo[6]);
69     let texto = document.getElementById("texto"+idtitulo[6]);
70     // console.log(texto);
71     // console.log(texto.style.display);
72     if (texto.style.display == "block") {
73         texto.style.display = "none";
74     } else {
75         texto.style.display = "block";
76     }
77 }

```

Para mostrarlo añadí una función aparece un observador(?) de eventos “onclick” como se ve en la siguiente foto:

```

50 |     div.innerHTML += "<h3 onclick='aparece(event)' id='"+`titulo${num_sugerencias}`+ titulo + "</h3>";
51 |     div.innerHTML += "<p id='"+`texto${num_sugerencias}`+` ` + texto + "</p>";
52 |     sugerencias.appendChild(div);

```

Defino la función *aparece* que recibe el evento. uso *.target* para obtener el elemento que “sufre(?)” el evento, en este caso el h3. Tomo su id y lo guardo en la variable *texto*.

La id del h3 y del texto contienen el mismo número luego puedo acceder al texto que corresponde al h3 sobre el que he pinchado fácilmente.

Si está en *style.display* tiene el valor “block”, entonces se está mostrando, luego lo oculto dándole el valor “none”. En caso contrario, es decir, tiene el valor “none”, significa que no se está mostrando, luego le doy valor “block” para que sí se muestre.

Decidí hacerlo con *<h3 onclick='aparece(event)'>* porque tuve el problema de que los párrafos se añaden de manera dinámica. Entonces si hace un queryselector, como se carga cuando se carga la página los elementos que quiero aún no existen. Con la solución de poner *<h3 onclick='aparece(event)'>* hace que como envía el evento puedo recibir el elemento HTML h3 mediante *.target*. Win for me 😊.

En esta sección no he añadido lo de cambiar la clase css al pasar el ratón por encima el ratón para no complicar más la cosa.

Possible mejora

Hacer una base de datos en la que guarde las sugerencias para que cuando se cierre el navegador no se pierdan.

Además, añadir la funcionalidad de eliminar sugerencias.

Consultas

Consulta_nombres

Introduces nombre y/o apellidos y en la tabla azul te salen los registros que concuerden.

Tomo el formulario en la variable *formulario* y espero que suceda el evento “*submit*” para aplicar una función. Primero, uso *preventDefault* para evitar que el navegador me lleve al documento php. Luego, tomo los datos del formulario con new FormData (en consola se puede ver las propiedades de dicho objeto).

```
▼ FormData ⓘ
  ▼ [[Prototype]]: FormData
    ► append: f append()
    ► delete: f delete()
    ► entries: f entries()
    ► forEach: f forEach()
    ► get: f ()
    ► getAll: f getAll()
    ► has: f has()
    ► keys: f keys()
    ► set: f ()
    ► values: f values()
    ► constructor: f FormData()
    ► Symbol(Symbol.iterator): f entries()
    ► Symbol(Symbol.toStringTag): "FormData"
  ► [[Prototype]]: Object
```

Hago una consulta mediante el método “POST” a *consulta_nombres.php* (el fetch).

```
1 let formulario = document.getElementById("formulario_nombre");
2 var respuesta = document.getElementById("contenido_nombre");
3
4 formulario.addEventListener("submit", function (e) {
5   e.preventDefault(); // Evita que se recargue la página
6
7   // let datos = new FormData(document.getElementById("formulario_nombre"))
8   let datos = new FormData(formulario);
9   // console.log(datos.get("nombre")); // Obtiene el valor del campo name
10  // console.log(datos.get("apellido")); // Obtiene el valor del campo name
11  console.log(datos)
12  console.log("hola")
13  fetch("consulta_nombres.php", {
14    method: "POST",
15    body: datos
16  })
```

El php recibe nombre y apellido y lo guarda en variables, establece la conexión mediante new PDO. Si el intento de conexión falla se devuelve “Error: (mensaje de error)”.

```

1  <?php
2  // Consulta a BD preparadas
3
4  //Obtenemos los datos del formulario
5  $nombre = $_POST['nombre'];
6  $apellido = $_POST['apellido'];
7
8  // Prueba por si en la web no funciona
9  // $nombre = 'Antonio';
10
11 // db, user y pass
12 $db = 'mysql:host=localhost;dbname=EconomiaDB;charset=utf8';
13 $user = 'root';
14 $pass = '';
15
16 // Conexión a BD
17 try {
18     $conn = new PDO($db, $user, $pass);
19 } catch (PDOException $e) {
20     echo "Error: " . $e -> getMessage();
21 }
22 //si lo que hay después del try no es exitoso,
23 //se ejecuta el catch y muestra el error que se produjo en la conexión (getMessage())
24
25 // Consulta a BD
26 $sql = "SELECT * FROM Cliente WHERE Nombre = :nombre OR Apellido = :apellido";
27

```

Hago la consulta preparada (uso `:algo` y `bindParam` en vez de poner directamente `$algo` en la consulta). `fetchAll` se usa para pasar de un array con subarrays (cada fila de datos de la consulta es un subarray) a un array asociativo.

```

// Resultados de la consulta en formato predeterminado (array de subarrays)
$resultadosDefault = array(
    array('John', 30),
    array('Alice', 25),
    array('Bob', 35)
);

```

ejemplo array predeterminado

```

// Resultados de la consulta en formato de array asociativo
$resultadosAsociativos = array(
    array('nombre' => 'John', 'edad' => 30),
    array('nombre' => 'Alice', 'edad' => 25),
    array('nombre' => 'Bob', 'edad' => 35)
);

```

ejemplo array asociativo

eb.

```
25 // Consulta a BD
26 $sql = "SELECT * FROM Cliente WHERE Nombre = :nombre OR Apellido = :apellido";
27
28 // Preparar la consulta
29 $consulta = $conn -> prepare($sql);
30
31 // Asignar valores a los parámetros
32 $consulta -> bindParam(':nombre', $nombre);
33 $consulta -> bindParam(':apellido', $apellido);
34
35 // Ejecutar la consulta
36 $consulta -> execute();
37
38
39 // Ceramos conexión a BD
40 $conn = null;
41
42 // Arreglo asociativo de la consulta
43 $resultado = $consulta -> fetchAll(PDO::FETCH_ASSOC);
44
45 // // Mostrar resultados
46 // while ($fila = $consulta->fetch(PDO::FETCH_ASSOC)) {
47 //     // $fila contiene los datos de la fila actual como un arreglo asociativo
48 //     echo "Nombre: " . $fila['Nombre'] . "<br>";
49 //     echo "Apellido: " . $fila['Apellido'] . "<br>";
50 //     // Puedes mostrar otros campos según la estructura de tu tabla
```

⌚ Watch SASS | Ln 56 Col 3 Spaces: 4 UTF-8 | F PHP 8.2 ↻

Hago json_encode y el echo.

Vuelvo al JS.

- El primer .then es una arrow function que espera tiene un argumento (res) al que se le aplica la función json que toma un documento en formato json y lo transforma a un objeto válido en JS.
- El siguiente .then
 1. Se borra la tabla en la que se visualizan los datos en el HTML, para que podamos visualizar la nueva consulta.
 2. Si la respuesta de la consulta no es vacía se añade una fila en la tabla del HTML por cada documento (fila de la consulta) del JSON.
Si la respuesta es vacía se colapsan las columnas y se hace sólo una fila con el mensaje “No se encontraron resultados”

```

16   .then(res => res.json())// promesas
17   .then(data => {
18     console.log(data);
19     respuesta.innerHTML = "";// Clear existing content
20
21     if (data.length > 0) { // Check if there are elements in the response
22       data.forEach(element => {
23         respuesta.innerHTML += `
24           <tr>
25             <td>${element.ID_Cliente}</td>
26             <td>${element.Nombre}</td>
27             <td>${element.Apellido}</td>
28             <td>${element.Fecha_nacimiento}</td>
29           </tr>
30         `;
31       });
32     } else {
33       console.log("No results found");
34       respuesta.innerHTML = "<tr><td colspan='4'>No se encontraron resultados</td></tr>";
35     }
36   })
37
38

```

Consulta_transacciones

Sobre nosotros Consulta nombres Consulta transacciones Insertar datos

Consulta Transacciones

Consulta las transacciones de una fecha en adelante y de una divisa concreta.

Fecha: dd/mm/yyyy <input type="button" value=""/>	<input type="checkbox"/> EURO: <input type="checkbox"/> Libra esterlina: <input type="checkbox"/> Franco suizo: <input type="checkbox"/> Zloty polaco: <input type="checkbox"/> Corona checa: <input type="checkbox"/> Corona noruega: <input type="checkbox"/> Corona sueca: <input type="checkbox"/>	<input type="button" value="Enviar"/> <input type="button" value="Reset"/>
---	--	--

ID_Transaccion	Fecha	Monto	Descripcion	ID_Cuenta	ID_Divisa

Contacto: info@bankconsulting.com

De qué trata explicado en [Página web](#).

Procedemos de manera parecida a consulta_nombres co el *addEventListener*. Tomamos en divisaSeleccionada todas las divisas que han sido seleccionadas. divisaSeleccionada es un nodeList, que aunque es un objeto parecido a un Array no lo es, así que lo convertimos en un array con Array.from y lo guardamos en divisaSeleccionada2.

La consulta que vamos a hacer es mediante el método get. Sería más apropiado hacerlo mediante el método post, ya que tenemos una array, pero vamos a tomarlo como un reto. Para solucionar el problema iteramos en el array y hacemos varias consultas get individualmente y añadimos a la tabla para visualizar datos. Borramos la tabla cada vez que le damos al botón del formulario.

```
1 var respuesta_transaccion = document.getElementById("contenido_transaccion");
2 let formulario_transaccion = document.getElementById("formulario_transaccion");
3
4
5
6 formulario_transaccion.addEventListener("submit", function (e) {
7     e.preventDefault(); // Evita que se recargue la página
8
9     let divisaSeleccionada = document.querySelectorAll('input[name="divisa"]:checked');
10    let divisaSeleccionada2 = Array.from(divisaSeleccionada);
11    console.log(divisaSeleccionada);
12    console.log(divisaSeleccionada2);
13    var fecha = document.getElementById("fecha");
14
15
16    let url = "http://localhost/bd2022/trabajo-base-datos/web/consulta_transaccion.php";
17    url = new URL(url);
18    url.searchParams.append("fecha", fecha.value);
19
20    // Borra el contenido de la tabla
21    respuesta_transaccion.innerHTML = ``;
22
23    // Por cada divisa hace una consulta
24    divisaSeleccionada2.forEach(divisa => {
25        console.log(divisa.value);
26        url.searchParams.append("divisa", divisa.value);
27        console.log(url.toString());
28    }
```

```
29 |     fetch(url, {method: "GET"})
30 |
31 |       .then(res => res.json())
32 |
33 |       .then(data => {
34 |         console.log(data);
35 |         data.forEach(element => {
36 |           respuesta_transaccion.innerHTML += `
37 |             <tr scope="row">
38 |               <td>${element.ID_Transaccion}</td>
39 |               <td>${element.Fecha}</td>
40 |               <td>${element.Monto}</td>
41 |               <td>${element.Descripcion}</td>
42 |               <td>${element.ID_Cuenta}</td>
43 |               <td>${element.ID_Divisa}</td>
44 |             </tr>
45 |           `
46 |         })
47 |       })
48 |     );

```

El php es una consulta get estándar salvo porque revisa que si no hay divisas seleccionadas hace la consulta teniendo en cuenta sólo la fecha, es decir, es como si hubiera seleccionado todas las divisas.

```

1  <?php
2  $fecha = $_GET["fecha"];
3  $divisa = $_GET["divisa"];
4
5  // $fecha = date("2020-01-01");
6  // $divisa = array("EUR", "GBP", "CHF");
7
8  // db, user y pass
9
10 $db = 'mysql:host=localhost;dbname=EconomiaDB;charset=utf8';
11 $user = 'root';
12 $pass = '';
13
14 // conexión a la BD
15 try {
16     $conn = new PDO($db, $user, $pass);
17 } catch (PDOException $e) {
18     echo "Error de conexión a la BD";
19     exit;
20 }
21 // Consulta a BD
22
23 if (empty($divisa)) {
24     $sql = "SELECT * FROM Transaccion WHERE Fecha >= :fecha";
25
26     // Preparar la consulta
27     $consulta = $conn -> prepare($sql);
28
29     // Asignar valores a los parámetros
30     $consulta -> bindParam(':fecha', $fecha);
31
32     // Ejecutar la consulta
33     $consulta -> execute();
34
35     echo json_encode($consulta -> fetchAll(PDO::FETCH_ASSOC));
36 } else {
37     $sql = "SELECT T.ID_Transaccion, T.Fecha, T.Monto, T.Descripcion, T.ID_Cuenta, T.ID_Divisa
38     FROM Transaccion as T INNER JOIN Divisa as D ON T.ID_Divisa = D.ID_Divisa
39     WHERE T.Fecha >= :fecha AND D.Codigo = :divisa";
40
41     // Preparar la consulta
42     $consulta = $conn -> prepare($sql);
43
44     // Asignar valores a los parámetros
45     $consulta -> bindParam(':fecha', $fecha);
46     $consulta -> bindParam(':divisa', $divisa);
47
48     // Ejecutar la consulta
49     $consulta -> execute();
50
51     // Solucionar posibles fallos de arrays no válidos a JSON
52     $resultado = $consulta->fetchAll(PDO::FETCH_ASSOC);
53     echo json_encode($resultado);
54 }
```

Insertar, eliminar datos

Voy a insertar clientes. Para ello inserto Nombre, Apellido y Fecha de nacimiento, además pongo ID_Cuenta que tiene, si no añado ningún ID_Cliente no se le asocia cuenta. La cuenta debe existir de antes.

Elimino clientes por el ID, si la cuenta que tenía asociada, se queda sin ningún cliente, elimino también la cuenta.

Problema surgido de array de arrays

Tras la siguiente consulta ([consulta16](#)):

```
$sql =  
"SELECT ID_Cuenta FROM Cliente_Cuenta  
WHERE ID_Cuenta IN  
(SELECT ID_Cuenta  
FROM Cliente_Cuenta  
GROUP BY ID_Cuenta  
HAVING COUNT(*) = 1)  
AND ID_Cliente = :ID_Cliente";  
  
$consulta = $conn->prepare($sql);  
$consulta -> bindParam(":ID_Cliente", $ID_Cliente);  
  
$consulta -> execute();  
  
  
$resultado = $consulta -> fetchAll(PDO::FETCH_ASSOC);
```

\$resultado queda como una array de arrays, como se puede ver con un vardump:

```
array(2) { [0]=> array(1) { ["ID_Cuenta"]=> int(9) } [1]=> array(1) { ["ID_Cuenta"]=> int(11) } }
```

(este vardump ha sido en un prueba.php y con unos datos que he añadido de prueba)

	ID_Cliente_Cuenta	ID_Cliente	ID_Cuenta
<input type="checkbox"/>	Edit Copy Delete	1	1
<input type="checkbox"/>	Edit Copy Delete	2	1
<input type="checkbox"/>	Edit Copy Delete	3	2
<input type="checkbox"/>	Edit Copy Delete	4	2
<input type="checkbox"/>	Edit Copy Delete	5	3
<input type="checkbox"/>	Edit Copy Delete	6	3
<input type="checkbox"/>	Edit Copy Delete	7	8
<input type="checkbox"/>	Edit Copy Delete	9	9
<input type="checkbox"/>	Edit Copy Delete	10	9
<input type="checkbox"/>	Edit Copy Delete	11	2
<input type="checkbox"/>	Edit Copy Delete	15	55
<input type="checkbox"/>	Edit Copy Delete	16	55

	ID_Cliente	Nombre	Apellido	Fecha_nacimiento
<input type="checkbox"/>	1	Juan	Pérez	1985-01-15
<input type="checkbox"/>	2	Ana	García	1990-06-22
<input type="checkbox"/>	3	Antonio	García	1970-01-01
<input type="checkbox"/>	4	Maximiliam	Müller	1990-01-01
<input type="checkbox"/>	5	Alexander	Zelensky	1986-01-01
<input type="checkbox"/>	6	Marilyn	Monroe	1926-01-01
<input type="checkbox"/>	7	Donald	Trump	1946-01-01
<input type="checkbox"/>	8	Jorge Mario	Bergoglio	1936-01-01
<input type="checkbox"/>	9	Dalia	Lama	1938-01-01
<input type="checkbox"/>	10	Vladimir	Putin	1952-01-01
<input type="checkbox"/>	11	Alex	elcapo	1987-10-06
<input type="checkbox"/>	12	Joanne	Rowling	1965-07-31
<input type="checkbox"/>	55	Daniel	Moreno	2000-06-28

Lo he solucionado accediendo al elemento (INT) mediante la key del array “ID_Cuenta”.

```

foreach ($resultado as $c) {
    // Eliminar de Cliente_Cuenta
    $sql = "DELETE FROM Cliente_Cuenta WHERE ID_Cuenta = :ID_Cuenta";
    $consulta = $conn->prepare($sql);
    $consulta -> bindParam("ID_Cuenta", $c["ID_Cuenta"]);
    $consulta -> execute();

    // Eliminar de Cuenta
    $sql = "DELETE FROM Cuenta WHERE ID_Cuenta = :ID_Cuenta";
    // Puedo poner solo ID_Cuenta porque ya he comprobado que esa cuenta solo tiene un cliente (el que va
    $consulta = $conn->prepare($sql);
    $consulta -> bindParam("ID_Cuenta", $c["ID_Cuenta"]);
    $consulta -> execute();
}

```