Solución reto Soporte - TI

Caso 1

```
"Hola chicos de soporte, El cliente con id: "65a3f2007a1b4f9d3c8e2b70", no le esta permitiendo crear una operación, podrian ayudarnos a saber que esta ocurriendo?"
```

Tareas

1. ¿Qué está sucediendo?

El usuario con ID "65a3f2007a1b4f9d3c8e2b70" (Pedro Rojas) no se le está permitiendo realizar más operaciones debido a que ha sobrepasado el límite de número operaciones para la categoría de su cuenta estándar, la cual tiene como límite 2 operaciones completadas por día.

2. Query de mongo

```
db.client.aggregate([
  {
    $match: {
      _id: ObjectId('65a3f2007a1b4f9d3c8e2b70')
    }
  },
  {
    $lookup: {
      from: 'operation',
      let: { clientId: '$_id' },
      pipeline: [
        {
          $match: {
            $expr: { $eq: ['$$clientId', '$clientId'] },
            date: {
              $gte: ISODate('2024-03-28T00:00:00Z'),
              $lt: ISODate('2024-03-28T23:59:59Z')
            }
          }
        }
      ],
      as: 'operations'
    }
  },
  {
    $addFields: {
      totalCompletedOperations: {
        $reduce: {
          input: '$operations',
          initialValue: 0,
          in: {
           $add: [
             '$$value',
             {
               $cond: [{ $eq: ['$$this.status', 2] }, 1, 0]
```

```
            }
          ],
        }
      }
    }
  }
])
```

EXPLICACIÓN

Esta consulta obtiene las operaciones realizadas por un cliente en base a una determinada fecha, esto es así debido a que las limitaciones de categoría son diarias por lo que quiero solo las de un día en específico, además agrego la propiedad "totalCompletedOperations" que facilitará el análisis de la información.

¿ERROR EN LA REGLA DE NEGOCIO O LÓGICA?

En este caso no habría error, ni en la regla de negocio ni lógica, ya que la razón por la que el cliente no puede hacer más operaciones es justamente porque se está haciendo respetar la regla de negocio para su tipo de cuenta.

3. Respuesta para el que reporto el problema

El usuario con ID: 65a3f2007a1b4f9d3c8e2b70 tiene una cuenta de categoría "standard", la cual tiene el límite de máximo 2 operaciones completadas al día. El usuario ya ha alcanzado ese límite y es debido a esto que no se le está permitiendo realizar más operaciones.

4. ¿Necesita abrir ticket?

No amerita ticket, ya que la razón de que el usuario no puede realizar más operaciones es que excedió el límite de operaciones para su tipo de cuenta, lo cual es el comportamiento deseado ya que respeta la regla de negocio existente.

Caso 2

```
"Tenemos un cliente con el siguiente id: 65a3f2007a1b4f9d3c8e2b71, que finalizo su operación con id: '65a3f1f87a1b4f9d3c8e2b62', estamos a punto de transferirle el valor de su operación, pero validamos que se le permitio crear una más allá de su límite de operaciones."
```

Tareas

¿Qué está sucediendo?

El cliente con ID "65a3f2007a1b4f9d3c8e2b71" (Laura Castro) tiene una cuenta de categoría premium, por lo que está sujeta a un máximo de 4 operaciones completadas al día, sin embargo ha podido realizar 5 operaciones en un día, superando su límite. Esto indica de que la regla de negocio no está siendo respetada.

Query

```
db.client.aggregate([
  {
    $match: {
      _id: ObjectId('65a3f2007a1b4f9d3c8e2b71')
    }
```

```
  },
  {
    $lookup: {
      from: 'operation',
      localField: '_id',
      foreignField: 'clientId',
      let: { clientId: '$_id' },
      pipeline: [
        {
          $match: {
            date: {
              $gte: ISODate('2024-03-28T00:00:00Z'),
              $lt: ISODate('2024-03-28T23:59:59Z')
            }
          }
        }
      ],
      as: 'operations'
    }
  },
  {
    $addFields: {
      totalCompletedOperations: {
        $reduce: {
          input: '$operations',
          initialValue: 0,
          in: {
           $add: [
             '$$value',
             {
               $cond: [{ $eq: ['$$this.status', 2] }, 1, 0]
             }
           ],
          }
        }
      }
    }
  }
])
```

Con este query puedo identificar que existe un problema al validar la regla de negocio, que debe ser visto y resuelto.

3. Respuestas

Efectivamente el cliente ha excedido su límite de operaciones diarias, probablemente la validación no se haya realizado o no ha funcionado correctamente, lo consultaré con desarrollo.

4. ¿Necesita ticket?

Sí, amerita ticket, debido a que una regla de negocio no ha sido respetada, permitiendo al cliente hacer una operación más de la que debería. Por lo que debe revisarse el proceso de validación al hacer cumplir esta regla.

Caso 3

```
Límites diarios reales
```

1. Query Mongo

```
db.client.aggregate([
  {
    $lookup: {
      from: 'operation',
      let: { clientId: '$_id' },
      pipeline: [
        {
          $match: {
            $expr: { $eq: ['$$clientId', '$clientId']},
            status: {
              $in: [1, 2]
            }
          }
        }
      ],
      as: 'operations'
    }
  },
  {
    $project: {
      operations: 1,
      name: 1,
      category: 1,
      totalAmount: {
        $sum: {
          $map: {
            input: '$operations',
            as: 'operation',
            in: {
              $cond: {
                if: {
                  $eq: [
                    '$$operation.currency',
                    'USD'
                  ]
                },
                then: {
                  $multiply: [
                    '$$operation.amountDestination',
                    3.78
                  ]
                },
                else: '$$operation.amountDestination'
              }
            }
```

```
        }
      }
    }
  }
},
{
  $addFields: {
    operations: {
      $map: {
        input: '$operations',
        as: 'operation',
        in: {
          $mergeObjects: [
            '$$operation',
            {
              requireReview: {
                $switch: {
                  branches: [
                    {
                      case: {
                        $and: [
                          {
                            $eq: [
                              '$$operation.currency',
                              'USD'
                            ]
                          },
                          {
                            $gt: [
                              '$$operation.amountDestination',
                              5500
                            ]
                          }
                        ]
                      },
                      then: true
                    },
                    {
                      case: {
                        $and: [
                          {
                            $eq: [
                              '$$operation.currency',
                              'PEN'
                            ]
                          },
                          {
                            $gt: [
                              '$$operation.convertedAmount',
                              5500
                            ]
                          }
```

```
                        ]
                      },
                      then: true
                    }
                  ],
                  default: false
                }
              }
            },
          ]
        },
      },
    },
  }
])
```

Caso 4

```
Actualiza todas las transacciones pendiente >48h a expirado, y ademas los clientes que tengan 2 o mas operaciones expiradas deben ser degradados a standard en su categoria.
```

Tareas

1. Query de mongo

ACTUALIZAR LAS OPERACIONES PENDIENTES > 48

```
db.operation.updateMany(
  {
    date: {
      $lt: new Date(Date.now() - 2 * 24 * 60 * 60 * 1000)
    },
    status: 1
  },
  {
    $set: {
      status: 4
    }
  }
)
```

NÚMERO DE OPERACIONES EXPIRADAS POR CLIENTE

```
db.operation.aggregate([
      {
        $group: {
          _id: "$clientId",
          totalExpiredOperations: {
            $sum: {
              $cond: [{ $eq: ["$status", 4] }, 1, 0],
            },
          },
        },
```

```
      },
      {
        $match: {
          totalExpiredOperations: {
            $gte: 2,
          },
        },
      },
      {
        $project: {
          _id: 1,
        },
      },
    ])
```

2. Script de automatización

```
const uri = "mongodb://localhost:27017/";
const client = new MongoClient(uri);
async function connect() {
  try {
    await client.connect();
    console.log("Connected successfully to server");
  } catch (err) {
    console.log(err);
  }
}
async function updateUserStatus() {
  await connect();
  const db = client.db("kambista");
  const clientCollection = db.collection("client");
  const operationCollection = db.collection("operation");

  const expiredOperations = await operationCollection.aggregate([
    {
      $match: {
        date: {
          $lt: new Date(Date.now() - 2 * 24 * 60 * 60 * 1000),
        },
        status: 1,
      }
    },
    {
      $project: {
        _id: 1,
        amountDestination: 1,
        currency: 1,
      }
    },
    { $addFields: { estadoExpirado: true } },
  ]).toArray();
  await operationCollection.updateMany(
    { _id: { $in: expiredOperations.map((operation) => operation._id) } },
```

```
  {
    $set: {
      status: 4
    }
  }
);
// ids clientes con operaciones expiradas (>2)
const clientsId = (await operationCollection
  .aggregate([
    {
      $group: {
        _id: "$clientId",
        totalExpiredOperations: {
          $sum: {
            $cond: [{ $eq: ["$status", 4] }, 1, 0],
          },
        },
      },
    },
    {
      $match: {
        totalExpiredOperations: {
          $gte: 2,
        },
      },
    },
    {
      $project: {
        _id: 1,
      },
    },
  ])
  .toArray()).map((client) => client._id);

const clientsBeforeUpdate = await clientCollection
  .find({
    _id: { $in: clientsId },
  }, { name: 1, category: 1 })
  .toArray();

await clientCollection.updateMany(
  { _id: { $in: clientsId } },
  {
    $set: {
      category: 'standard'
    }
  }
)

return {
  expiredOperations,
  updatedClients: clientsBeforeUpdate.map(client => ({
```

```
      id: client._id,
      name: client.name,
      nivelAnterior: client.category,
      nivelActual: 'standard'
    }))
  };
}
```

RESPUESTA

```
{
  expiredOperations: [
    {
      _id: new ObjectId('65a3f1f87a1b4f9d3c8e2b03'),
      amountDestination: 4000,
      currency: 'USD',
      estadoExpirado: true
    },
    {
      _id: new ObjectId('65a3f1f87a1b4f9d3c8e2b04'),
      amountDestination: 8000,
      currency: 'PEN',
      estadoExpirado: true
    },
    {
      _id: new ObjectId('65a3f1f87a1b4f9d3c8e2b05'),
      amountDestination: 9000,
      currency: 'PEN',
      estadoExpirado: true
    }
  ],
  updatedClients: [
    {
      id: new ObjectId('65a3f2007a1b4f9d3c8e2b91'),
      name: 'Albert Hammond',
      nivelAnterior: 'premium',
      nivelActual: 'standard'
    }
  ]
}
```

EXPLICACIÓN

Esta script actualiza las operaciones pendientes > 48 horas a expiradas, luego actualiza los usuarios con 2 o más operaciones expiradas y lo hace de la siguiente manera:

1. Me conecto a la base de datos y selecciono las colecciones a usar
2. Primero obtengo las operaciones que ya han expirado (pendientes > 48 h), seleccionando solo algunas propiedades y agrego una nueva (estadoExpirada)
3. Luego actualizo el "status" de estas operaciones a 4 (expirado), mediante el uso de map para solo extraer los ids de las operaciones y usarlo como condición.

4. Teniendo las operaciones expiradas, obtengo los clientes que tengan 2 o más de estas, seleccionando solo su ID.
5. Obtengo los datos de estos usuarios, para saber que su "category" inicial, antes de la actualización.
6. Actualizo el "category" de los clientes obtenidos a 'standar'.
7. Finalmente retorno las operaciones expiradas y uso map para retornar un arreglo que muestre el nivel anterior y actual del cliente.

Caso 5

Tareas

1. Query para identificar transacciones afectadas

```
db.operation.aggregate([
  {
    $match: {
      currency: 'PEN',
      convertedAmount: {
        $gt: 5500
      },
      manualReview: {
        $exists: false
      }
    }
  }
])
```

2. Solución temporal

Para solucionar este problema realizaría los siguientes pasos:

- Primero indentificaría las operaciones afectadas
- Identificar cuál de esas operaciones aún no han sido completadas para validarlas antes que pasen al área de liquidaciones.
- Dar aviso al área de operaciones y liquidaciones sobre las operaciones completadas no revisadas y comunicarse con los clientes asociados a esas operaciones.

3. Propuesta fix

PROBLEMA

Las transacciones en 'PEN' no están siendo identificadas para la revisión manual para cuando supera su equivalente a los 5,500 USD, debido a que la conversión se aplica después de la validación inicial.

Base de datos del problema: Kambista - colección operation

SOLUCIÓN

```
db.operation.aggregate([
  {
    $match: {
      currency: 'PEN',
    }
  },
  {
    $addFields: {
      needManualReview: {
```

```
        $cond: [
          { $gt: ['$convertedAmount', 5500] },
          true,
          false
        ]
      }
    }
  }
])
```

Esta consulta, permite obtener las operaciones en 'PEN' y agregar una propiedad que indica si debe ser revisada manualmente (neeManualReview).

Con este script se busca indentificar las operaciones con una currency 'PEN' para validarlas y saber si necesitan una revisión manual en base a su equivalente en dólares.

Así el usuario no tendrá inconvenientes a la hora de realizar sus operaciones, como también el área de operaciones y liquidaciones sabrán que operaciones deben ser validadas.

Caso 6

```
Incidentes simultáneos
```

Tareas

1. Ordena las soluciones y justifica tu respuesta.

Para dar solución a todos estos incidentes primero priorizaría cada uno en base a su relevancia y en base a ello realizaría las siguientes acciones:

1. Asegurarme del correcto funcionamiento de la base de datos

Primero debo enfocarme en la base de datos, ya que para los otros incidentes necesitaré la información almacenada en la misma para analizarlos adecuadamente y encontrar las causas que originaron el incidente. Además de que la BD debe estar activa para que los usuarios puedan interactuar con la plataforma y esta este funcionando correctamente.

Plan de acción para darle solución:

- Visualizar los logs de mongo y entender que está sucediendo, ¿Cuál es la causa del problema?
- Una vez haya indentificado una causa o causas las indagaría para econtrar una solución, usando documentación oficial, foros, etc.
- Aplicar las soluciones y asegurarme de que la base de datos este funcionando correctamente.

2. Atender a los usuarios VIP

En este caso los usuarios VIP tienen el mayor rango de categoría por lo que debería darles mayor atención, además de que ya han realizado operaciones completadas y estás aún no se ven reflejadas, por lo que internamente hay un problema que está impidiendo las transacciones, y este problema podría afectar a todos los usuarios, no solo a los

VIP, afectando al negocio core de Kambista (transacciones de cambio de divisa) y su marca.

Una vez la base de datos esté funcionando, realizaría lo siguiente:

- Identificar las transacciones afectadas
- Comunicarme con el área de liquidaciones para obtener información sobre estas transacciones e identificar los problemas
- Generar soluciones para los problemas encontrados

3. Falla de script de actualización

Cuando la base de datos esté funcionando y las transacciones estén siendo efectuadas, tocaría analizar la razón del porque el script de actualización falló.

Plan de acción:

- Descomponer el script en partes más pequeñas para ver que sucede en cada parte del script
- Analizar cada parte específica del script, realizar pruebas para validar el comportamiento esperado e identificar bugs.
- Con los bugs indentificados les daría una solución para que el script se comporte como debe
- Comprobar cada parte del script corregido y validar su funcionamiento con datos de prueba.

2. ¿Qué información pedirías a cada área involucrada o con que personas te comunicarías?

Mi comunicaría principalmente con el área de liquidaciones y operaciones, para indentificar la razón del porque las transacciones no están siendo reflejadas, buscaría pedir información acerca del estado de las transacciones para saber si estas realmente ya han sido completadas, de ser el caso obtener la información del porque aún no se refleja la operación. También vería si los datos entregados o consultas están generando el problema, realizaría preguntas como:

- ¿Se ha realizado la transacción exitosamente?
- ¿Hay algún inconveniente con los datos que esté impidiendo la transacción?

Teniendo identificados los posibles problemas buscaría soluciones para cada caso desde las capacidades de mi puesto, si una de estos problemas está más del lado de desarrollo crearía un ticket que contenga información sobre el problema, la información encontrada que esté relacionada y posibles soluciones.