Implementando Verificação de Duplicidade na Função addTransaction

1: Atualizar a Função addTransaction no Controlador

1. **Abra o arquivo transactionsController.js dentro da pasta controllers**.
2. **Atualize a função addTransaction**:

Código JavaScript

```
const addTransaction = (req, res) => {
  const { date, amount, description, category, account, user_id } = req.body;

  // Verificar se a transação já existe
  db.query(
    'SELECT * FROM transactions WHERE date = ? AND amount = ? AND description = ? AND category = ? AND account = ? AND user_id = ?',
    [date, amount, description, category, account, user_id],
    (err, results) => {
      if (err) {
        console.error('Erro ao verificar transação:', err);
        res.status(500).send('Erro ao verificar transação');
        return;
      }

      if (results.length > 0) {
        // Se a transação já existe
        res.status(400).send('Transação duplicada');
        return;
```

```
    }


    // Se a transação não existe, insira-a no banco de dados
    db.query(
      'INSERT INTO transactions (date, amount, description, category, account, user_id) VALUES (?, ?, ?, ?, ?, ?)',
      [date, amount, description, category, account, user_id],
      (err, results) => {
        if (err) {
          console.error('Erro ao adicionar transação:', err);
          res.status(500).send('Erro ao adicionar transação');
          return;
        }
        res.status(201).send('Transação adicionada com sucesso');
      }
    );
  }
 );
};
```

Explicação do Código:

1. **Consulta SQL**: A consulta SQL SELECT * FROM products WHERE name = ? AND description = ? AND expiry_date = ? verifica se já existe um produto no banco de dados com o mesmo name, description e expiry_date.
2. **Callback da Consulta**: A função de callback recebe dois parâmetros: err (para erros) e results (para os resultados da consulta).
3. **Verificação de Erro**: Se err não for nulo, significa que houve um erro ao executar a consulta, e uma mensagem de erro é enviada ao cliente.
4. **Verificação de Duplicidade**:
    - results.length > 0: Se essa condição for verdadeira, significa que a consulta encontrou um produto existente com os mesmos valores fornecidos. Portanto, a função envia uma resposta de erro 400 (Bad Request) indicando "Produto duplicado".
    - results.length === 0: Se essa condição for verdadeira (implícita), significa que a consulta não encontrou produtos duplicados, e a função continua para adicionar o novo produto ao banco de dados.

Conclusão

A condição results.length > 0 é crucial para a verificação de duplicidade. Ela assegura que a API não insere registros duplicados no banco de dados, mantendo a integridade e consistência dos dados.

Parte Prática (1 hora):

Ação:

1. **Atualizar a função addTransaction no arquivo transactionsController.js para incluir a verificação de duplicidade**.
2. **Testar a rota POST para /api/transactions para garantir que a duplicidade é tratada corretamente**.

Teste Prático com Insomnia:

Vamos usar o Insomnia para testar a rota POST para /api/transactions e verificar a duplicidade.

Testando a Rota POST para Adicionar Transações com Verificação de Duplicidade

1. **Abra o Insomnia**.
2. **Crie uma Nova Requisição**:
    - No menu lateral esquerdo, clique no botão "+" ao lado de "Debug" para adicionar uma nova requisição.
    - Dê um nome à sua requisição, por exemplo, "Add Transaction".
    - Selecione o método HTTP como "POST".
    - Clique em "Create".
3. **Configure a URL da Requisição**:
    - Na barra de URL, insira http://localhost:3000/api/transactions.
4. **Configure o Corpo da Requisição**:
    - Na aba "Body", selecione "JSON".
    - Adicione o seguinte JSON:

Código JSON

```
{
  "date": "2023-07-08",
  "amount": 300.00,
  "description": "Viagem",
```

```
  "category": "Lazer",
  "account": "Cartão de Crédito",
  "user_id": 1
}
```

5. **Envie a Requisição**:
    - Clique no botão "Send" para enviar a requisição.
6. **Verifique a Resposta**:
    - Verifique o painel de resposta no Insomnia.
    - Se a transação já existe, a resposta deve indicar "Transação duplicada".
    - Se a transação não existe, a resposta deve indicar "Transação adicionada com sucesso".

Exemplo de Respostas Esperadas:

Se a Transação Já Existe

Código JSON

```
{
  "message": "Transação duplicada"
}
```

Se a Transação Não Existe

Código JSON

```
{
  "message": "Transação adicionada com sucesso"
}
```

Resolução de Problemas Comuns:

Erro 404 (Not Found)

- Verifique se o servidor está rodando corretamente.
- Certifique-se de que a URL está correta e que a rota /api/transactions está configurada no server.js.

Erro 500 (Internal Server Error)

- Verifique os logs do servidor para identificar a causa do erro.
- Certifique-se de que a conexão com o banco de dados está configurada corretamente.

Atualizando o Controlador para Verificar se a Transação Existe

1. **Atualizar a Função updateTransactionPut**

Abra o arquivo **transactionsController.js** dentro da pasta **controllers**.

Atualize a função **updateTransactionPut**:

Código JavaScript

```
const updateTransactionPut = (req, res) => {
  const { id } = req.params;
  const { date, amount, description, category, account, user_id } = req.body;
  db.query(
    'UPDATE transactions SET date = ?, amount = ?, description = ?, category = ?, account = ?, user_id = ? WHERE id = ?',
    [date, amount, description, category, account, user_id, id],
    (err, results) => {
      if (err) {
        console.error('Erro ao atualizar transação:', err);
        res.status(500).send('Erro ao atualizar transação');
        return;
      }
      if (results.affectedRows === 0) {
        res.status(404).send('Transação não encontrada');
```

```
      return;
    }
    res.send('Transação atualizada com sucesso');
  }
 );
};
```

2. Atualizar a Função updateTransactionPatch

Atualize a função **updateTransactionPatch**:

Código JavaScript

```
const updateTransactionPatch = (req, res) => {
 const { id } = req.params;
 const fields = req.body;
 const query = [];
 const values = [];

 for (const [key, value] of Object.entries(fields)) {
  query.push(`${key} = ?`);
  values.push(value);
 }

 values.push(id);

 db.query(
  `UPDATE transactions SET ${query.join(', ')} WHERE id = ?`,
  values,
  (err, results) => {
   if (err) {
```

```
      console.error('Erro ao atualizar transação:', err);
      res.status(500).send('Erro ao atualizar transação');
      return;
    }
    if (results.affectedRows === 0) {
      res.status(404).send('Transação não encontrada');
      return;
    }
    res.send('Transação atualizada com sucesso');
  }
 );
};
```

3. Atualizar a Função deleteTransaction

Atualize a função **deleteTransaction**:

Código JavaScript

```
const deleteTransaction = (req, res) => {
 const { id } = req.params;
 db.query('DELETE FROM transactions WHERE id = ?', [id], (err, results) => {
  if (err) {
   console.error('Erro ao deletar transação:', err);
   res.status(500).send('Erro ao deletar transação');
   return;
  }
  if (results.affectedRows === 0) {
   res.status(404).send('Transação não encontrada');
   return;
  }
  res.send('Transação deletada com sucesso');
```

```
  });
};
```

Explicação do Código:

A condição if (results.affectedRows === 0) é usada para verificar se a operação de atualização ou deleção de um registro no banco de dados não afetou nenhuma linha.

Veja em detalhes o que isso significa e como funciona no contexto de uma função que atualiza ou deleta um registro.

O que a Condição if (results.affectedRows === 0) Faz

1. **Execução da Consulta de Atualização ou Deleção**:
    - Quando uma consulta SQL de atualização (UPDATE) ou deleção (DELETE) é executada, o banco de dados retorna um objeto de resultado que contém informações sobre a operação, incluindo o número de linhas afetadas (affectedRows).
2. **Verificação de Linhas Afetadas**:
    - results.affectedRows retorna o número de linhas que foram afetadas pela consulta.
    - results.affectedRows === 0 verifica se nenhuma linha foi afetada pela consulta.

Significado da Condição

- **Se results.affectedRows === 0 for verdadeiro**:
    - Isso significa que a consulta não encontrou nenhum registro que corresponda aos critérios especificados (por exemplo, um id que não existe na tabela).
    - No contexto de uma operação de atualização ou deleção, isso indica que o registro que se tentou atualizar ou deletar não foi encontrado no banco de dados.
- **Se results.affectedRows === 0 for falso**:
    - Isso significa que a consulta afetou uma ou mais linhas.
    - No contexto de uma operação de atualização ou deleção, isso indica que o registro foi encontrado e a operação foi realizada com sucesso.

Testando as Funções Atualizadas

Vamos usar o Insomnia para testar as rotas PUT, PATCH e DELETE novamente, garantindo que retornem a mensagem apropriada caso a transação não exista.

Testando a Rota PUT para Atualizar Transações

1. **Abra o Insomnia**.
2. **Crie uma Nova Requisição**:
    - Dê um nome à sua requisição, por exemplo, "Update Transaction (PUT)".
    - Selecione o método HTTP como "PUT".
    - Configure a URL da requisição para http://localhost:3000/api/transactions/999 (onde 999 é um ID que não existe).
    - Na aba "Body", selecione "JSON" e adicione o seguinte JSON:

Código JSON

```
{
  "date": "2023-07-08",
  "amount": 300.00,
  "description": "Viagem atualizada",
  "category": "Lazer",
  "account": "Cartão de Crédito",
  "user_id": 1
}
```

    - Clique no botão "Send" para enviar a requisição.
    - Verifique a resposta no Insomnia.

Testando a Rota PATCH para Atualizar Transações

1. **Crie uma Nova Requisição**:
    - Dê um nome à sua requisição, por exemplo, "Update Transaction (PATCH)".
    - Selecione o método HTTP como "PATCH".
    - Configure a URL da requisição para http://localhost:3000/api/transactions/999 (onde 999 é um ID que não existe).
    - Na aba "Body", selecione "JSON" e adicione o seguinte JSON:

Código JSON

```
{
  "amount": 400.00,
  "description": "Viagem parcialmente atualizada"
}
```

- Clique no botão "Send" para enviar a requisição.
- Verifique a resposta no Insomnia.

Testando a Rota DELETE para Deletar Transações

1. **Crie uma Nova Requisição**:
    - Dê um nome à sua requisição, por exemplo, "Delete Transaction".
    - Selecione o método HTTP como "DELETE".
    - Configure a URL da requisição para http://localhost:3000/api/transactions/999 (onde 999 é um ID que não existe).
    - Clique no botão "Send" para enviar a requisição.
    - Verifique a resposta no Insomnia.

Exemplo de Resposta Esperada quando a Transação Não Existe

Código JSON

```
{
  "message": "Transação não encontrada"
}
```