1. **Escopo do projeto**

Solicita-se o desenvolvimento de uma api que controle os gastos de clientes.

Este projeto será utilizado para registrar clientes e seus gastos. O software da Tria Software servirá como entregador de dados visando a elaboração de relatórios para tomadas de decisões.

**Produtos e Serviços**

* Desenvolvimento de Api utilizando NodeJS

Devem ser cadastrados os clientes que entraram em contato, para isso se faz necessário gravar

* Id, (number)
* Nome da empresa, (string)
* Nome completo, (string)
* Telefone (string)
* E-mail (string)
* Gastos

Devem ser cadastrados os gastos que essa empresa teve diariamente

* Id, (number)
* Valor, (number) – not integer
* Data (YYYY-MM-DD hh:mm:ss) – string

END POINTS DA API

( Os testes da sua API podem ser feitos através da ferramenta INSOMNIA ou softwares equivalentes que possam apresentar o retorno dos resultados )

* Através de uma requisição, apresentar uma lista com os todos os clientes e seus gastos.
* Através de uma requisição com parâmetros, apresentar um determinado cliente e seus gastos.
* Através de uma requisição com parâmetros, apresentar um determinado cliente e seus gastos filtrados por um período de data (start date, end date).
* Através de uma requisição com parâmetros, apresentar um determinado cliente, seus gastos, a soma total dos seus gastos e a média diária de gastos de um determinado mês
* Ao ser passado um JSON OBJECT no body de requisição, podendo ser um cliente ou apenas gasto, se o ID existir no banco de dados, os dados devem ser atualizados.
* Ao requisitar um ID de um gasto ou de um cliente, o mesmo deve ser excluído do banco de dados.
* Ao ser passado um JSON OBJECT no body de requisição, podendo ser um cliente ou apenas gasto, mesmo deve ser salvo no banco de dados.

**Tecnologias exigidas**

O sistema deve ser construído utilizando NodeJS.  
Para o banco de dados, deve ser utilizado PostgreSQL.

As classes utilizadas no sistema devem ser corretamente anotadas utilizando TypeORM. Todos os campos da classe devem ser anotados, assim como o relacionamento entre elas existente.

**DESAFIOS**

1. A API deve ser construída visando atender requisições em GraphQL.
2. Procure respeitar os tipos de exceptions lançadas, assim como os retornos HTTP(200,404,500 etc).
3. O banco de dados deve estar em um ambiente Docker e separado da aplicação.
4. No banco de dados Postgres, instale e utilize a extensão TimeScaleDB.
5. Para a construção e modificações das tabelas no banco de dados, utilize arquivos de migration.
6. Utilizando a mensageria RabbitMQ, ao postar mensagens em filas diferentes, deve ser possível criar, editar, ou até mesmo excluir clientes ou gastos. ( Para este desafio o RabbitMQ deve ser utilizado em um ambiente Docker, separado da aplicação )
7. Tenha amor pelo seu código, utilize uma boa estrutura, padrões e tenha um código limpo.
8. Crie um Readme de como rodar e consultar sua API.