Stone Tech | Desafio

O desafio é dividido em 2 etapas.
Na primeira nós queremos ver o seu conhecimento em bancos de dados. Você escreverá um código SQL puro para criar tabelas em um banco de dados relacional qualquer e uma query.
Na segunda nós queremos ver seu conhecimento em programação. Você terá que escrever um código resolvendo um problema e o subir no github.
Você pode resolver o teste usando uma das seguintes linguagens: Java, PHP, JavaScript, C#, Python, Ruby, Kotlin, Go, Erlang, Elixir, Rust ou C.
Você não precisa se preocupar com performance e conhecimento de bibliotecas na resolução do teste. Esses e outros aspectos abordaremos durante a sua formação conosco. Se preocupe em fazer uma solução que seja clara e objetiva, ou seja, não inclua funcionalidades além do que pedimos.
O prazo para realizar o teste fica a seu critério. Quando finalizar, envie para **Tatiana Matera** (Tati) com o link do github da sua solução e as respostas das perguntas de Banco de Dados.

Banco de dados
1. Escreva um SQL para criar tabelas em um banco de dados relacional qualquer (ex: PostgreSQL, MySQL e etc) para uma aplicação bancária que terá contas e transações.
- Essas contas devem ter número da agência e número da conta.
- Essas transações são depósitos, saques, transferências para outros bancos (agência, conta e documento), recebimento de transferências de outros bancos e pagamentos de boletos.
- Escreva também código SQL que cria índices básicos para garantir a integridade dos dados.

2. Escreva uma query SQL para retornar o valor em dinheiro total de pagamentos de boletos de cada mês do ano atual.
- Escreva também código SQL que cria um índice adequado para essa query.

Programação
Imagine uma lista de compras. Ela possui:
- Itens
- Quantidade de cada item
- Preço por unidade de cada item

Desenvolva uma função (ou método) que receberá uma lista de compras (como a detalhada acima) e uma lista de e-mails. Aqui, cada e-mail representa uma pessoa.
A função deve:
- Validar que não exista algum e-mail duplicado.
- Validar que nenhuma das listas esteja vazia.
- Calcular a soma dos valores, ou seja, multiplicar o preço de cada item por sua quantidade e somar todos os itens.
- Validar que a quantidade e valor de um item não sejam negativos.
- Dividir o valor de forma igual entre a quantidade de e-mails.

- Retornar um mapa/dicionário onde a chave será o e-mail e o valor será quanto ele deve pagar nessa conta.
- Ter testes unitários.

IMPORTANTE!

- **Não trabalhe com valores com decimais**. Ou seja, ponto flutuante ou float. Use inteiros para representar os valores! Como a menor unidade na nossa moeda é o centavo, use valores inteiros em centavos. Assim, um real será representado por 100 (cem centavos é igual a um real).
- **Divida o valor de forma igual!** Por exemplo, um valor total de 102 (R$ 1,02) para ser dividido entre 4 e-mails. O final deve ser duas pessoas com 26 e duas pessoas com 25.
- Quando fizer a divisão, **é importante que não falte nenhum centavo**! Cuidado quando tiver, por exemplo, um valor total de 100 (R$ 1,00) para ser dividido entre 3 e-mails. Isso daria uma dízima infinita. No entanto, o correto aqui é que duas pessoas fiquem com o valor 33 e uma fique com 34.
- Mantenha a representação de valores em números inteiros até o fim, ou seja, **os valores finais não devem ser representados ou convertidos para números de ponto flutuante.**
- O resultado da função deve ser um mapa/dicionário. Em javascript essas estruturas de dados são conhecidas como objetos.
- Escreva testes unitários para verificar as validações de e-mails duplicados, listas vazias e total maior que zero.
- **Escreva testes unitários** para verificar os cenários possíveis de divisão. Como o valor total de 100 para 3 pessoas, de 102 para 4 pessoas e o de 1 para 3 pessoas.
- Não trabalhe com arquivos, apenas envie para a função as duas listas.
- **Deixe documentado como você executa os testes unitários da solução**. Por exemplo: "Eu instalo a versão X da linguagem, eu compilo o código com o comando Y e executo os testes unitários com o comando Z".

Mensagem final para você!

Divirta-se no processo! Aproveite esse pequeno teste como uma prática divertida de problemas de programação. **Esse pode ser um pontapé inicial para acontecimentos maiores na sua carreira! ;D**