Desafio Técnico — API de Tabela Tarifária de Água

Visão geral

Desenvolver uma API REST para gerenciar e calcular tarifas de água com base em categorias de consumidores e faixas de consumo. O sistema deve ser totalmente parametrizável, permitindo ajustes nas tarifas sem alteração de código.

Stack Tecnológica

- **Linguagem**: Java
- **Framework**: Spring Boot
- **Banco de Dados**: PostgreSQL
- **Arquitetura**: API REST com retorno JSON

Conceitos e Regras de Negócio

1.1 Categorias de Consumidores

O sistema deve suportar pelo menos quatro categorias distintas de consumidores:

- **COMERCIAL**: Estabelecimentos comerciais
- **INDUSTRIAL**: Indústrias e fábricas
- **PARTICULAR**: Residências
- **PÚBLICO**: Órgãos públicos

Requisito importante: Cada categoria deve ter vínculo claro e persistente com suas respectivas faixas de consumo e valores dentro de uma tabela tarifária.

1.2 Faixas de Consumo

A cobrança é feita por faixas progressivas de consumo, cada uma com seu valor unitário (R$/m³).

Exemplo de estrutura de faixas:

```
Faixa 1: 0 a 10 m³
Faixa 2: 11 a 20 m³
Faixa 3: 21 a 30 m³
Faixa 4: 31 a 99999 m³
```

Critério de Parametrização: Todas as faixas e seus valores devem ser armazenados no banco de dados, permitindo ajustes futuros sem modificação de código.

1.3 Regras de Validação e Consistência

Para garantir a integridade da tabela tarifária, as seguintes regras devem ser aplicadas:

Regra	Descrição
Não sobreposição	Faixas não podem ter intervalos que se cruzam
Ordem válida	O valor inicial de cada faixa deve ser menor que o valor final (início < fim)
Cobertura completa	Deve iniciar em 0 (zero) m³
Cobertura suficiente	Deve haver faixas que cubram qualquer consumo informado

Funcionalidades da API

2.1 Gerenciamento de Tabelas Tarifárias

Criar Tabela Tarifária

`POST /api/tabelas-tarifarias`

Funcionalidade: Cria uma nova tabela tarifária completa com todas as categorias e suas respectivas faixas de consumo.

Entrada: JSON contendo a estrutura completa da tabela (categorias + faixas + valores)

Listar Tabelas Tarifárias

`GET /api/tabelas-tarifarias`

Funcionalidade: Retorna todas as tabelas tarifárias cadastradas no sistema.

Saída: Lista de tabelas com suas respectivas configurações

Excluir Tabela Tarifária

`DELETE /api/tabelas-tarifarias/{id}`

Funcionalidade: Remove uma tabela tarifária do sistema.

Parâmetro: `id` - Identificador único da tabela

Importante: Deve impedir que tabelas excluídas sejam utilizadas em cálculos futuros.

2.2 Cadastro em Lote via JSON

O sistema deve aceitar cadastro completo de tabelas tarifárias através de um payload JSON estruturado.

Responsabilidade do Desenvolvedor: Propor o modelo/schema do JSON baseado na estrutura de banco de dados criada.

2.3 Cálculo do Valor a Pagar

Endpoint de Cálculo

`POST /api/calculos`

Entrada (Request Body):

json

```
{
  "categoria": "INDUSTRIAL",
  "consumo": 18
}
```

Regra de Cálculo

O cálculo é feito **progressivamente** por faixas:

Consumo em cada faixa × Valor unitário da faixa = Subtotal da faixa

Exemplo Prático:

Categoria: Industrial

Consumo Total: 18 m³

Tabela de Faixas:

- Faixa 1: 0 a 10 m³ → R$ 1,00/m³
- Faixa 2: 11 a 20 m³ → R$ 2,00/m³

Cálculo Detalhado:

1. **Faixa 1** (0-10m³): 10 m³ × R$ 1,00 = **R$ 10,00**
2. **Faixa 2** (11-20m³): 8 m³ × R$ 2,00 = **R$ 16,00**

Valor Total: R$ 10,00 + R$ 16,00 = **R$ 26,00**

Formato de Retorno Obrigatório

O retorno deve ser **detalhado** e incluir:

json

```
{
  "categoria": "INDUSTRIAL",
  "consumoTotal": 18,
  "valorTotal": 26.00,
  "detalhamento": [
    {
      "faixa": {
        "inicio": 0,
        "fim": 10
      },
      "m3Cobrados": 10,
      "valorUnitario": 1.00,
      "subtotal": 10.00
    },
    {
      "faixa": {
        "inicio": 11,
        "fim": 20
      },
      "m3Cobrados": 8,
      "valorUnitario": 2.00,
      "subtotal": 16.00
    }
  ]
}
```

Campos obrigatórios:

- `categoria`: String com a categoria utilizada
- `consumoTotal`: Integer com o consumo informado
- `valorTotal`: BigDecimal com o valor total a pagar
- `detalhamento`: Array com breakdown por faixa
  - `faixa.inicio`: Integer
  - `faixa.fim`: Integer
  - `m3Cobrados`: Integer (quantidade de m³ cobrados naquela faixa)
  - `valorUnitario`: BigDecimal
  - `subtotal`: BigDecimal

Modelagem de Dados

Requisitos de Modelagem

O desenvolvedor deve propor uma estrutura relacional que suporte:

1. **Entidade Principal**: `TabelaTarifaria`
   - Representa uma tabela tarifária completa
   - Pode ter campos como: id, nome, data de vigência, etc.
2. **Vínculo por Categoria**: Relacionamento claro entre tabela e categoria
3. **Faixas Parametrizadas**: Armazenamento de faixas vinculadas à categoria e à tabela
4. **Parametrização Total**: Mudanças em faixas/valores devem refletir automaticamente nos cálculos, sem alteração de código

---

Critérios de Aceite

Para que o desafio seja considerado completo, a solução deve demonstrar:

☐ **Criar** tabela de tarifária completa (todas as categorias, faixas e seus relacionamentos)

☐ **Listar** todas as tarifas cadastradas por faixa de acordo com a categoria fornecida

☐ **Calcular** valor correto para um consumo específico, respeitando:

- Categoria informada
- Faixas cadastradas
- Cálculo progressivo por faixas

☐ **Demonstrar parametrização**: Alterações nas faixas/valores no banco devem refletir automaticamente nos cálculos, sem necessidade de alterar código

---

Entregáveis

O repositório deve conter:

1. Código Fonte

- Implementação completa da API
- Organização clara de pacotes/camadas
- Código limpo e bem estruturado

2. README.md

Deve incluir:

- **Instruções de instalação e execução**
- **Pré-requisitos** (versões do Java, PostgreSQL, etc.)
- **Configuração do banco de dados**
- **Exemplos de requests e responses** para cada endpoint
- **Como testar a aplicação**

3. Scripts de Banco de Dados

- Script SQL de criação das tabelas
- Ou configuração equivalente (ex: migrations, JPA schema generation)
- Opcional: script com dados de exemplo (seed)