Documentação do Projeto - State CSV Batch & Aggregator

1. Visão Geral

Este sistema é composto por dois projetos principais:

1. State CSV Batch Processor
   - Lê um arquivo CSV com registros de telefone.
   - Agrupa os dados por estado.
   - Envia os lotes para o serviço agregador via HTTP.

2. State Aggregator Service
   - Recebe os lotes populacionais por estado.
   - Agrega e armazena no banco de dados.
   - Exponibiliza endpoints para consulta e gerenciamento dos dados.

2. State CSV Batch Processor

Localização: c:/Users/Wictorsama/source/desafio/state-csv-batch-processor

Objetivo:
- Ler e agrupar registros do arquivo src/data/phone_data.csv.
- Enviar os dados em lotes para o endpoint /states/batch do serviço agregador.
- Logar os dados processados e ignorados.

Arquivo principal: src/main.ts

Componentes:
- CsvReaderService: lê e processa linha a linha.
- Lógica de Agrupamento: agrupa em lotes de 1000 registros (configurável).
- Cliente HTTP: usa axios para enviar os lotes.
- Logging: registra status e contagem de registros.

Funcionamento:
1. Inicia a aplicação NestJS.
2. Lê o arquivo CSV.
3. Para cada registro válido, adiciona ao lote.
4. Ao atingir o limite, envia o lote.
5. Ao final, envia o último lote e registra totais.

Exemplo de CSV:

id,name,phone,state
1,John Doe,(11) 99999-9999,SP

Exemplo de lote enviado:

```
{
  "states": [
    { "name": "SP", "population": 1 },
    { "name": "RJ", "population": 1 }
  ]
}
```

3. State Aggregator Service

Localização: c:/Users/Wictorsama/source/desafio/state-aggregator-service

Objetivo:
- Receber, agregar e armazenar dados populacionais por estado.
- Oferecer API REST para consulta e gerenciamento dos dados.

Controller principal: src/presentation/controllers/state.controller.ts

Endpoints:
- POST /states/batch: recebe lote de estados.
- GET /states/aggregates: retorna dados agregados por estado.
- GET /states/count: retorna contagem total de registros.
- DELETE /states/all: remove todos os registros.

Exemplo de requisição:

```
{
  "states": [
    { "name": "SP", "population": 1 },
    { "name": "RJ", "population": 1 }
  ]
}
```

Exemplo de resposta:

```
{
  "success": true,
  "processedStates": 2,
  "aggregatedData": [
    { "state": "SP", "population": 1 },
```

```
    { "state": "RJ", "population": 1 }
  ]
}
```

Resposta de /states/aggregates:

```
[
  { "state": "SP", "totalPopulation": 1234 },
  { "state": "RJ", "totalPopulation": 567 }
]
```

Banco de Dados:
- MongoDB: armazena os dados agregados por estado.
- Cada documento representa um estado com sua população.

4. Execução dos Projetos

Pré-requisitos:
- Node.js e npm instalados.
- MongoDB e Redis em execução (pode usar Docker Compose).

Passos:
1. Iniciar MongoDB e Redis:
   docker-compose up -d mongodb redis

2. Iniciar Aggregator Service:
   cd c:/Users/Wictorsama/source/desafio/state-aggregator-service
   npm install
   npm run start:dev

3. Iniciar Batch Processor:
   cd c:/Users/Wictorsama/source/desafio/state-csv-batch-processor
   npm install
   npm run start:dev

4. Acessar Swagger:
   http://localhost:3001/api

5. Solução de Problemas

- Sem dados em /states/aggregates:

  - Verifique se ambos os serviços estão em execução.
  - Verifique os logs.
  - Confirme se há documentos na coleção 'states' do MongoDB.

- CSV não totalmente processado:
  - Verifique logs de registros processados e ignorados.
  - Confirme o caminho e permissão do arquivo.

6. Extensões Possíveis

- Ajustar o tamanho dos lotes em src/main.ts (BATCH_SIZE).
- Adicionar validações e transformações adicionais.
- Criar novos endpoints e recursos no serviço agregador.