

Prova técnica – Desenvolvedor Delphi / Python

0. Visão geral

Neste desafio você vai:

1. Criar uma conta no nosso backend de candidatos (Supabase).
2. Confirmar seu e-mail.
3. Fazer login e obter um **ACCESS_TOKEN** (JWT).
4. Implementar um programa (Delphi ou Python) que:
 - lê um `input.csv` com municípios e populações;
 - enriquece os dados usando a **API de localidades do IBGE**;
 - gera um `resultado.csv`;
 - calcula estatísticas;
 - envia automaticamente essas estatísticas para uma API de correção (Edge Function), usando o seu **ACCESS_TOKEN**.

Tempo estimado: **1h30 a 2h**.

1. Cadastro e login (obrigatório)

Nosso backend do candidato está no Supabase:

- **SUPABASE_URL**:
`https://mynxlubykylncttggu.supabase.co`
- **SUPABASE_ANON_KEY** (API Key pública – pode usar no seu código):
`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi0iJzdXBhYmFzZSIiLnJlZiI6Im15bnhsdWJ5a3lsbmNpbmR0Z2d1Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NjUxODg2NzAsImV4cCI6MjA4MDc2NDY3MH0.Z-zqiD6_tjnF2WLU167z7jT5NzZaG72dWH0dpQW1N-Y`

Você pode interagir com o Auth via HTTP (curl, Postman, código, etc.).

1.1 Criar usuário (signup)

Faça um **POST** para:

NASAJON

None

<https://mynxlubykylncinttggu.supabase.co/auth/v1/signup>

Exemplo (substitua pelo SEU e-mail e SUA senha):

None

```
curl -X POST "https://mynxlubykylncinttggu.supabase.co/auth/v1/signup" \
-H "Content-Type: application/json" \
-H "apikey:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIssInJlZiI6Im15bnh
sdWJ5a3lsbmNpbnR0Z2d1Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NjUxODg2NzAsImV4cCI6MjA
4MDc2NDY3MH0.Z-zqiD6_tjnF2WLU167z7jT5NzZaG72dWH0dpQW1N-Y" \
-d '{
  "email": "SEU_EMAIL_AQUI",
  "password": "SUA_SENHA_FORTE_AQUI",
  "data": {
    "nome": "Seu Nome Completo"
  }
}'
```

Importante:

- Use um e-mail ao qual você tenha acesso (vai chegar o link de confirmação).
- Guarde sua senha – você usará no login.

1.2 Confirmar o e-mail

Você receberá um e-mail do Supabase.

Clique no link de confirmação.

Após o clique, você será redirecionado para uma página do “Portal de Candidatos” (GitHub Pages da Nasajon) com uma mensagem do tipo:

“Conta confirmada com sucesso! Bem-vindo, Seu Nome!”

Se não receber o e-mail:

- verifique a pasta de spam;
- use a funcionalidade de "[reenviar e-mail de confirmação](#)" no portal

1.3 Fazer login e obter o ACCESS_TOKEN

Depois de confirmar a conta, faça login usando:

POST

`https://mynxlubykylncttggu.supabase.co/auth/v1/token?grant_type=password`

Exemplo:

```
None
curl -X POST
"https://mynxlubykylncttggu.supabase.co/auth/v1/token?grant_type=password"
\
-H "Content-Type: application/json" \
-H "apikey:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi0iJzdXBhYmFzZSI
sdWJ5a3lsbmNpbnR0Z2d1Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NjUxODg2NzAs
4MDc2NDY3MH0.Z-zqiD6_tjnF2WLU167z7jT5NzZaG72dWH0dpQW1N-Y" \
-d '{
    "email": "SEU_EMAIL_AQUI",
    "password": "SUA_SENHA_FORTE_AQUI"
}'
```

A resposta será algo como:

```
None
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....",
  "token_type": "bearer",
  "expires_in": 3600,
  "user": {
    "id": "uuid-do-usuario",
```

```
"email": "SEU_EMAIL_AQUI",
"user_metadata": {
    "nome": "Seu Nome Completo"
}
}
```

Guarde o valor de **access_token** – este será o seu **ACCESS_TOKEN**, que você usará no header:

Authorization: Bearer <ACCESS_TOKEN>

dentro do seu programa, quando for enviar os resultados para a API de correção.

2. Arquivo de entrada (input.csv)

Use exatamente o seguinte arquivo:

```
municipio,populacao
Niteroi,515317
Sao Gonçalo,1091737
Sao Paulo,12396372
Belo Horizonte,2530701
Florianopolis,516524
Santo Andre,723889
Santos Andre,700000
Rio de Janeiro,6718903
Curitiba,1963726
Brasilia,3094325
```

3. API do IBGE (localidades)

Use a API pública de localidades do IBGE:

- Documentação: <https://servicodados.ibge.gov.br/api/docs/localidades>

Você pode, por exemplo:

- Fazer um GET geral:
`https://servicodados.ibge.gov.br/api/v1/localidades/municipios`
e montar uma estrutura em memória,
- Ou usar outra estratégia desde que consiga, para cada município de entrada, obter:
 - nome oficial,
 - UF,
 - região,
 - código IBGE.

A lógica de matching (tratamento de acentos, maiúsculas/minúsculas, erros de digitação, etc.) fica a seu critério.

4. Arquivo de saída (resultado.csv)

Seu programa deve gerar um `resultado.csv` com as colunas:

`municipio_input, populacao_input, municipio_ibge, uf, regiao, id_ibge, status`

Onde:

- `municipio_input` : nome original do `input.csv`.
- `populacao_input` : valor original do `input.csv`.
- `municipio_ibge` : nome oficial retornado pelo IBGE.
- `uf` : sigla da unidade federativa.
- `regiao` : região (Norte, Nordeste, Centro-Oeste, Sudeste, Sul).
- `id_ibge` : código numérico do município.
- `status` : um destes valores:
 - OK
 - NAO_ENCONTRADO
 - ERRO_API
 - AMBIGUO (se você optar por tratar casos com múltiplos matches).

5. Estatísticas que devem ser calculadas

Depois de processar todos os municípios, seu programa deve calcular:

1. `total_municipios`
2. `total_ok`
3. `total_nao_encontrado`
4. `total_erro_api`
5. `pop_total_ok`
 - soma das populações (`populacao_input`) de linhas com `status = "OK"`.
6. `medias_por_regiao`
 - média de `populacao_input` por região (considerando apenas status OK), por exemplo:

JSON

```
"medias_por_regiao": {  
    "Sudeste": 999999.17,  
    "Sul": 999999.0,  
    "Centro-Oeste": 999999.0  
}
```

6. Envio dos resultados para a API de correção

Ao final da execução, seu programa deve:

1. Montar um JSON neste formato:

JSON

```
{  
    "stats": {
```

```
"total_municipios": 99,  
"total_ok": 99,  
"total_nao_encontrado": 99,  
"total_erro_api": 99,  
"pop_total_ok": 99999,  
"medias_por_regiao": {  
    "Sudeste": 999999.17,  
    "Sul": 999999.0,  
    "Centro-Oeste": 999999.0  
}  
}  
}
```

(Os valores acima são apenas um exemplo de estrutura, não os números reais.)

2. Fazer um **POST** para a nossa Edge Function de correção.

Exemplo genérico de chamada :

```
JSON  
PROJECT_FUNCTION_URL="https://mynxlubykylncttggu.functions.supabase.co/ibg  
e-submit"  
ACCESS_TOKEN="SEU_ACCESS_TOKEN_AQUI"  
  
curl -X POST "$PROJECT_FUNCTION_URL" \  
-H "Authorization: Bearer $ACCESS_TOKEN" \  
-H "Content-Type: application/json" \  
-d '{  
    "stats": {  
        "total_municipios": 99,  
        "total_ok": 9,  
        "total_nao_encontrado": 9,  
        "total_erro_api": 0,  
        "pop_total_ok": 99999,  
        "medias_por_regiao": {  
            "Sudeste": 999999.7,  
            "Sul": 99999.0,  
            "Centro-Oeste": 99999.0  
        }  
    }  
}'
```

```
    }  
}'
```

No seu programa (Delphi ou Python), você deverá:

- montar o JSON de **stats** com os valores calculados;
- enviar para a URL que informaremos;
- incluir o header **Authorization: Bearer <ACCESS_TOKEN>**;
- ler a resposta e **imprimir a nota (score) no console.**

A resposta tem o formato:

JSON

```
{  
  "user_id": "uuid...",  
  "email": "seu_email@exemplo.com",  
  "score": 8.75,  
  "feedback": "Muito bom! Seu resultado está bem próximo do gabarito.",  
  "components": { ... }  
}
```

8. O que será avaliado

- Correção dos dados gerados em **resultado.csv**.
- Precisão das estatísticas calculadas.
- Qualidade e organização do código.
- Tratamento de erros (API fora, município não encontrado, etc.).
- Clareza e honestidade nas explicações das decisões.

- Implementação correta do fluxo de autenticação (login) e envio à API de correção com o **ACCESS_TOKEN**.

O uso de IA é permitido, mas:

- você deve entender o código que está entregando;
- poderá ser convidado a explicar detalhes da solução numa conversa técnica.

9. Entrega

Envie (copy cole no final deste texto)

- código-fonte (Delphi ou Python);
- `input.csv` (o de entrada);
- `resultado.csv` (gerado pelo seu programa);
- **Notas Explicativas** - explicando as principais decisões técnicas
- se necessário, um `README` muito curto explicando como rodar (versão do Delphi ou comando `python main.py`, etc.).

Boa prova 😊

COLE EMBAIXO OS SEUS ARTEFACTOS OU O LINK DO SEU REPOSITÓRIO GITHUB (se preferir)