# Senha válida

## Prova Backend

Joaquim é um jovem estudante que está fazendo junto com sua classe um sistema para sua faculdade. Cada aluno foi responsável por uma parte do sistema e Joaquim está responsável pela parte do sistema que verifica a força da senha.
Tendo em vista que o sistema pode ter vários níveis de acesso diferentes, a API que Joaquim deve produzir precisa trabalhar com vários conjuntos de regras.

## O Problema

Dada uma palavra contínua, e um conjunto de regras, Joaquim precisa verificar se a senha é válida baseada nas regras pedidas.

Regras possíveis:

- `minSize`: tem pelo menos **x** caracteres.
- `minUppercase`: tem pelo menos **x** caracteres maiúsculos
- `minLowercase`: tem pelo menos **x** caracteres minúsculos
- `minDigit`: tem pelo menos **x** dígitos (0-9)
- `minSpecialChars`: tem pelo menos **x** caracteres especiais ( Os caracteres especiais são os caracteres da seguinte string: "`!@#$%^&*()-+\/{}[]`" )
- `noRepeted`: não tenha nenhum caractere repetido em **sequência** ( ou seja, "aab" viola esta condição, mas "aba" não)

## Entrada / Saída

Para facilitar a consulta dos amigos de Joaquim, você deve receber a string por meio de uma API Web. Ou seja, cada consulta é feita através de uma requisição HTTP que deve retornar o resultado em formato JSON. A sua API pode ser feita seguindo os padrões REST ou GraphQL. Você pode escolher o formato que se sentir mais confortável. A entrada é sempre válida.

A professora do Joaquim costuma dar pontos extras para APIs em formato GraphQL :)

Apenas para a regra “noRepeted” não teremos um valor configurável, então em nossos testes vamos mandar como 0 mas pode ser ignorado.

#### API REST

A API deve possuir uma única rota **/verify** que recebe uma requisição REST em formato JSON contendo a senha e uma lista de regras. Por exemplo:

**URL:** http://localhost:8080/verify
**Method**: POST

```
{
    "password": "TesteSenhaForte!123&",
    "rules": [
        {"rule": "minSize","value": 8},
        {"rule": "minSpecialChars","value": 2},
        {"rule": "noRepeted","value": 0},
        {"rule": "minDigit","value": 4}
    ]
}
```

A resposta deve ser feita também em formato JSON, e deve retornar um mapa com duas chaves:

- **verify**: que deve retornar um boolean dizendo se a senha foi validada por todas as regras
- **noMatch**: que deve retornar uma lista de strings que deve conter quais as regras a senha não passou ou uma lista vazia caso **verify** seja true.

Segue abaixo um exemplo de retorno  para o exemplo da requisição acima.

**Content-Type**: application/json

```
{
   "verify": false,
   "noMatch": ["minDigit"]
}
```

**API GraphQL**

A API GraphQL deve possuir uma única rota **/graphql** que recebe uma requisição contendo uma única *query* chamada **verify**. Por exemplo, para a entrada apresentada anteriormente, a requisição seria:

**URL:** http://localhost:8080/graphql
**Method**: POST

```
query {
    verify(password: "TesteSenhaForte!123&", rules: [
      {rule: "minSize",value: 8},
      {rule: "minSpecialChars",value: 2},
      {rule: "noRepeted",value: 0},
      {rule: "minDigit",value: 4}
    ]) {
        verify
        noMatch
    }
}
```

## Observações

- ➔ Deixe claro como seu programa deve ser executado. Scripts de automatização, testes e Dockerfiles são sempre bem-vindos.
- ➔ Documente sua lógica de implementação para entendermos o máximo possível do seu programa.
- ➔ Considere que as entradas serão sempre válidas.

**Boa prova =)**