Prova de Desenvolvimento Backend

Competências Avaliadas

Criar Controladores/Endpoints
Configurar uma rota personalizada
Configurar um método HTTP personalizado
Receber dados pela URL
Receber dados pelo Corpo
Receber os dados por Query
Receber os dados pela Header
Retornar o Status Code
Reconhecer qual o melhor Status Code a usar
Retornar um JSON
Criar um serviço e configurá-lo
Utilizar um serviço definido
Configurar um serviço com tempo de vida
Reconhecer o melhor tempo de vida do serviço
Criar um modelo JPA
Criar um repositório JPA
Definir uma query personalizada em um repositório JPA
Modificar e salvar dados dos modelos JPA
Configurar o Cors
Aplicar o BCrypt no armazenamento de senhas
Configurar o JWT para autentificação
Configurar o Spring Security
Criar um middleware
Saber estruturar um projeto organizado e com sentido
Saber resolver problemas usando desenvolvimento Backend
Aplicar o princípio da Responsabilidade Única (S)
Aplicar o princípio do Aberto-Fechado (O)
Aplicar o princípio da Segregação de Interface (I)
Aplicar o princípio da Inversão de Dependência (D)
Criar testes com JUnit
Criar testes que façam sentido e sejam úteis
Representar relações nos modelos JPA
Acessar dados usando relações dos modelos JPA

Questões

Você foi contratado por um governo de um país jovem a construir o sistema de endereçamento postal semelhante ao brasileiro, isto é, baseado em um Código de Endereçamento Postal (CEP). A estrutura do código que será usada por esse país é VV-EECC-RRRRRRRR-DD.

- Os dígitos V são a versão, que será usada caso o CEP sofra modificações futuramente, então, o valor sempre será 01, mas considere que seu software possa sofrer modificações futuras.
- Os dígitos E são o código do estado.
- Os dígitos C são o código da cidade.
- Os dígitos R são o código residencial. Atribuídos aleatoriamente.
- Os dígitos D são dígitos de verificação. Eles devem ser iguais a soma de todos os dígitos anteriores, exceto a versão, módulo de 100.

O seu serviço deverá ter, então os endpoints abaixo. Considere os seguintes parâmetros de qualidade:

- Retorne sempre mensagens corretas e com status code adequado.
- Não retorne dados sensíveis ou Ids do banco de dados usando DTO's.
- Você deve organizar o código separando em serviços e usando interfaces para melhor organização, organize seus controladores de forma que façam sentido.
- Também siga todos os padrões de segurança como JWT e BCrypt para senha.
- Quando solicitado, o JWT sempre será mandado pelo Header no campo Authorization no formato 'Bearer tokenjwtaqui'.
- Configure o CORS para uma url do governo: https://www.cepservice.gov.
- Utilize um Filter (middleware) para autentificação JWT.
- Extra: Faça testes com JUnit para verificar suas validações de CEP.
- Extra: Modifique seu sistema para permitir uma relação entre Cidade e Estado.

POST /user

```
{
   "username": [NOME DE USUARIO]
   "password": [SENHA DO USUARIO]
}
```

Cadastra um usuário no sistema. O usuário terá um código com vários caracteres aleatórios gerados para ele. Retorne esse código no campo *usercode* além de um código *message* com informações da requisição.

POST /auth

{
  “username”: [NOME DE USUARIO]
  “password”: [SENHA DO USUARIO]
}
Loga na conta de um usuário retornando um JWT no campo *token* e um campo *message* com a mensagem de sucesso/erro da operação.

POST /admin
{
  “usercode”: [USERCODE DE OUTRO USUARIO]
}
[Requer Autentificação][Apenas para Admin] Se o usuário dono do JWT for administrador do sistema, concede status de administrador ao usuário dono do *usercode* enviado.

GET /state?page=0&count=-1
[Requer Autentificação] Retorna um JSON com todos os estados limitados a quantidade expressa na variável *count* e na página expressa pela variável *page*. Se *count* é -1, retornar todos os estados. O estado deve retornar com seu nome e código.

GET /state/count
[Requer Autentificação] Retorna um JSON com o campo *count* que conta quantos estados estão cadastrados. Além disso um campo *nextCode* que retorna o código do próximo estado que será cadastrado. Dica: É bom armazenar no banco essas informações, já que estados podem ser deletados. Para contar utilize uma query personalizada com a instrução COUNT.

POST /state
{
  “state”: [NOME DO ESTADO],
  “code”: [CÓDIGO DO ESTADO]
}
[Requer Autentificação][Apenas para Admin] Cadastra um novo estado no banco de dados com o código apontado. Não deve cadastrar um mesmo estado duas vezes nem mesmo ter um código repetido.

PUT /state/{id}
{
  “state”: [NOME DO ESTADO],

"code": [CÓDIGO DO ESTADO]
}
[Requer Autentificação][Apenas para Admin] Modifica integramente os dados de um estado dado um Id.

PATCH /state/{id}
{
"state": [NOME DO ESTADO],
"code": [CÓDIGO DO ESTADO]
}
[Requer Autentificação][Apenas para Admin] Modifica apenas os campos enviados, não alterando os dados não enviados que serão nulos.

DELETE /state/{id}
[Requer Autentificação][Apenas para Admin] Deleta um estado. Por requisitos do governo do país, você não deve deletar ceps atrelados aquele estado. Além disso, o código do estado deletado não pode ser reutilizado.

GET /city/count
[Requer Autentificação] Retorna um JSON com o campo *count* que conta quantas cidades estão cadastrados. Além disso um campo *nextCode* que retorna o código da próxima cidade que será cadastrado. Dica: É bom armazenar no banco essas informações, já que cidades podem ser deletadas. Para contar utilize uma query personalizada com a instrução COUNT.

POST /city
{
"city": [NOME DO CIDADE],
"code": [CÓDIGO DO CIDADE]
}
[Requer Autentificação][Apenas para Admin] Cadastra uma nova cidade no banco de dados com o código apontado. Não deve cadastrar uma mesma cidade duas vezes nem mesmo ter um código repetido.

GET /cep/{cep}
Retorna um JSON com a rua, bairro, cidade, estado, e versão do cep. Tratar CEPs inválidos.

POST /cep
{

"rua": [RUA],
"bairro": [BAIRRO],
"cidade": [CIDADE],
"estado": [ESTADO]
}

[Requer Autentificação][Apenas para Admin] Cadastra um novo CEP retornando um JSON com o campo "code" com o código CEP do lugar. Para isso você precisará procurar no banco os nomes das cidades e estados e retornar seu código além de gerar um código residencial ainda não usado e calcular seus dígitos verificadores.