

Operadores de Mutação – Spring Security

Este documento organiza variações e mutações possíveis sobre expressões de autorização em Spring Security, considerando **roles** e **authorizations**.

1 `hasAnyRole(...)` / `hasAnyAuthorization(...)`

Caso base (original)

```
hasAnyRole("User", "Admin", "Guest")
```

1.1 Roles inexistentes / negativas

```
hasAnyRole("No_User", "No_Admin", "No_Guest")
hasAnyRole("No_User", "No_Admin", "Guest")
hasAnyRole("No_User", "Admin", "Guest")
hasAnyRole("User", "No_Admin", "No_Guest")
hasAnyRole("User", "Admin", "No_Guest")
hasAnyRole("User", "No_Admin", "Guest")
```

1.2 Roles alternativas existentes (caso situacional)

Quando a aplicação possui **outras roles/authorizations** além das esperadas

```
hasAnyRole("OutraExistente", "OutraExistente", "OutraExistente")
hasAnyRole("OutraExistente", "OutraExistente", "Guest")
hasAnyRole("OutraExistente", "Admin", "Guest")
hasAnyRole("User", "OutraExistente", "OutraExistente")
hasAnyRole("User", "Admin", "OutraExistente")
hasAnyRole("User", "OutraExistente", "Guest")
```

1.3 Variação na quantidade de parâmetros

```
hasAnyRole("User", "Admin")
hasAnyRole("User")
```

```
hasAnyRole("Admin", "Guest")
hasAnyRole("Guest")
hasAnyRole("Admin")
hasAnyRole()
```

1.4 Acesso irrestrito / bloqueado

```
permitAll()
denyAll()
```

1.5 Troca de tipo (Role ↔ Authorization)

```
hasAnyAuthorization("User", "Admin", "Guest")
```

2 hasRole(...) / hasAuthorization(...)

Caso base

```
hasRole("User")
```

2.1 Role inexistente

```
hasRole("No_User")
```

2.2 Role alternativa existente

```
hasRole("OutraExistente") // para as N outras roles da aplicação
```

2.3 Ausência de parâmetro

```
hasRole()
```

2.4 Acesso irrestrito / bloqueado

```
permitAll()  
denyAll()
```

2.5 Troca de tipo

```
hasAuthorization("User")
```

3 permitAll() como ponto de mutação

Caso base

```
permitAll()
```

3.1 Restringindo acesso

```
denyAll()
```

3.2 Substituição por controle explícito

```
hasRole("QualquerUmaExistente") // para todas as N roles  
hasAuthorization("QualquerUmaExistente") // para todas as N authorizations
```

3.3 Controle múltiplo

```
hasAnyRole(...)           // permutando quantidade e valores  
hasAnyAuthorization(...)  // idem
```



Classificação dos Operadores de Mutação

- **Mutação de valor:** troca de roles/authorizations
- **Mutação estrutural:** alteração na quantidade de parâmetros
- **Mutação semântica:** troca por `permitAll` / `denyAll`
- **Mutação de modelo de segurança:** Role ↔ Authorization