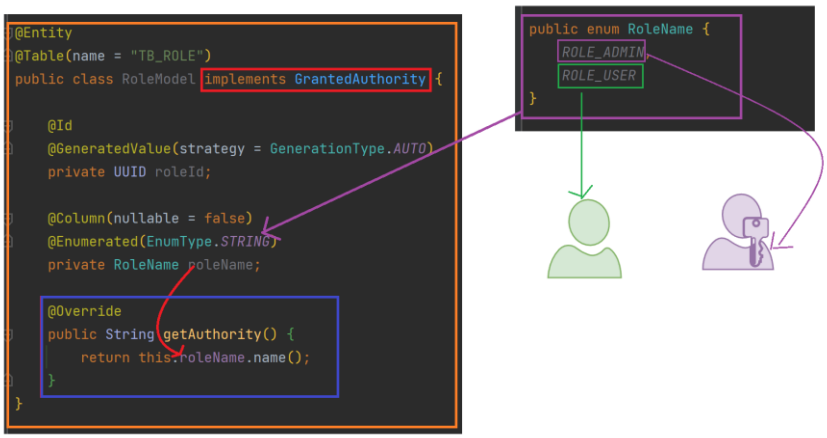
Autorização baseada em Roles - Spring Security

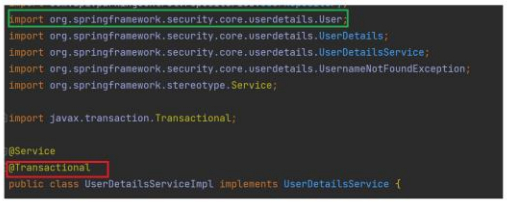
Numa aplicação **é comum que usuários tenham papéis diferentes**, pode ser um usuário free, outro pago, um admin etc.... Usuários com papéis/roles diferentes tem acesso a recursos de maneiras diferentes, para ajudar nessa implementação o **Spring Security** oferece um recurso de roles.

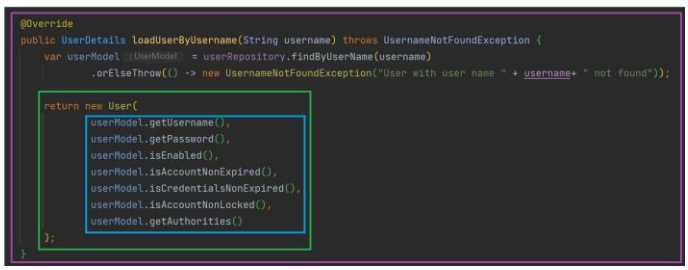
Primeiro precisamos de **uma entidade que represente as Roles disponíveis da aplicação** *(isso depende do contexto)*, essa entidade então **vai implementar a interface GrantedAuthority** que vai **simplesmente retornar a role:**



Esses **papéis precisam ser relacionados ao usuário**, são vários papéis que vários usuários podem ter, tornando a relação **ManyToMany** com uma **tabela de join simples**:

Uma outra mudança bem vinda **é quando retornamos o usuário na autenticação**, podemos **usar uma classe de User do próprio Spring Security** (essa classe também é um UserDetails) para envelopar o **modelo de usuário**, esse **User** é **construído com as propriedades do usuário de domínio**:





O uso da **anotação Transactional** nessa classe permite que **as roles sejam carregadas corretamente**, isso porque elas são lazy *(o que significa que só serão buscadas quando forem ser utilizadas)*, então **o Spring Security ao perceber que precisa carregar essas informações volta ao banco de dados e faz uma busca**, isso só **é possível pois o escopo transacional continua aberto através da anotação.**

Controlando endpoints por Roles - Spring Security

Uma vez que os usuários possuem papéis e estão devidamente associados a eles, podemos melhorar a segurança da aplicação restringindo determinados endpoints.

Um dos modos de configurar restrição/permissão é através do HttpSecurity, podemos restringir um **método HTTP de determinado verbo** com **determinado caminho**, tornando-o **acessível a determinado role**:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .httpBasic() HttpBasicConfigurer<HttpSecurity>
            .and() HttpSecurity
            .authorizeHttpRequests() AuthorizeHttpRequestsConfigurer<...>.AuthorizationManagerRequestMatcherRegistry
            .antMatchers(HttpMethod.GET, ...antPatterns: "/parking-spot/**").permitAll()
            .antMatchers(HttpMethod.POST, ...antPatterns: "/parking-spot").hasRole(RoleName.ROLE_USER.name())
            .antMatchers(HttpMethod.DELETE, ...antPatterns: "/parking-spot").hasRole(RoleName.ROLE_ADMIN.name())
            .anyRequest().authenticated()
            .and() HttpSecurity
            .csrf().disable();
}
```