Curso Spring Boot API REST

Criação do projeto no spring.start.io -> tudo ok

1. Habilitando segurança com **Spring Security** (passos)

* Adicionar a dependência **spring-boot-start-security** no arquivo **pom.xml**
* No pacote config.security, criar a classe **SecurityConfigurations**
* Habilitar esta classe com as anotações **@EnableWebSecurity e @Configuration**
* Esta classe tbm deve extender **WebSecurityConfigurerAdapter**
* Somente com estas alterações, o spring boot, por default, já bloqueia todo acesso a todos os endpoints da aplicação. Retornando um código **401 Unauthorized.**

1. **Liberando acesso aos endpoints do tipo GET da aplicação**

Na classe **SecurityConfigurations,** devemosimplementar os três métodos herdados da classe **WebSecurityConfigurerAdapter.**

1. **Restringindo acesso aos endpoints da aplicação**

* Neste processo é preciso autenticar o usuário, através da tela de login do próprio spring.
* Iremos utilizar o email e senha do usuário para autenticar.
* Para isso devemos informar o spring que a classe usuário será usada para autenticação. Para isso a **classe** usuário deve implementar a **interface UserDetails, do próprio spring boot** e implementar os métodos desta interface.
* Esta **classe usuário** deve ter também uma lista de perfis, para isso é preciso criar a **classe Perfil**, e fazer dela uma tabela no banco de dados, para futuramente fazer autorização do usuário conforme o perfil do usuário.
* Esta **classe Perfil,** também deve implementar uma **interface GrantedAuthoroty** e implementar o método.
* Não deve se esquecer de mapear a cardinalidade entre a **entidade Usuario e Perfil ManyToMany(Fetch.EAGER).**
* No método configure da classe **SecurityConfigurations,** que tem o parâmetro HttpSecurity, deve habilitar a tela de login do próprio **Spring.**
* Pronto com essas configurações temos acesso a tela de login.

1. **Autenticando o usuário**

* Editar o método **Configure** que recebe o parâmetro **AuthenticationManagerBuilder.**
* Criar a **classe AutenticacaoService** com a anotação **@Service,** e também deve implementar a **interface UserDetailsService,** consequentemente implementar o **método loadUserByUsername(String username).** Este método é o responsável por buscar o usuario no banco.
* O spring automaticamente faz a checagem no banco de dados para fazer a autenticação.
* Necessário criar a **classe UsuarioRepository,** para fazer uso do método **FindByEmail(String email)**
* **Não esquecer de encriptar a senha do usuário no banco.**
* Até aqui, esta forma de autenticação tradicional, gera uma sessão, ou seja, quando o usuário loga pela primeira vez, o servidor gerará um id (cuck) para cada usuário e envia para o cliente. Quando o usuário logar novamente o servidor com o id gerado e guardado tbm no servidor, confirma a autenticação. Não é uma boa prática, porque caso tiver milhões de usuário, isso vai consumir processamente e memória no servidor.

1. **Autenticar via token**

* Adicionar a dependência jjwt no arquivo pom.xml
* Desabilitar a tela de login do spring no método **configure** da **classe SecurityConfigurations.**
* **Também é preciso criar a classe AutenticacaoController, com a anotação @RestController e @RequestMapping(“/auth”)**
* **Criar o método post autenticar(*@RequestBody @Valid* FormLogin form)**
* **Para isso é preciso crizar a classe LoginForm, com apenas os atributos email e senha, com seus getters e setters.**