**SPRING SECURITY**

* **JWT:**
  + Ele tem um mecanismo muito simples e robusto de manter as informações de manter as informações do usuário por meio de um token q vai ser trafegado no cabeçalho das requisições. Basicamente vamos armazenar o **user** e o **tempo de expiração do token** de uma forma criptografada e assinada pelo nosso back-end. Então outra pessoa não consegue gerar um token valido pq este token **é assinado,** existe uma **palavra secreta** q ela vai ser embaralhada junto com o nosso token e isso faz o token ficar seguro.
* **Spring Security:** framework especifico do spring para segurança
* **Dependências :**
  + Jjwt
  + spring-boot-starter-security
* **classes:**
  + **SecurityConfiguracao extends WebSecurityConfigurerAdapter**
* **METODOS**
  + **cors()** se tiver o corsConfigurationSource definido então as suas configurações serão aplicadas chamando o método cors. Tem o objeto de liberar o acesso para todos os servidores
* **PESQUISAR**
  + **Ataque csrf():**  no projeto está desativo pq nosso sistema é stateless então não precisamos nos preocupar com este tipo de ataque q é baseado no armazenamento de autenticação e sessão.como não armazenaremos sessão, não tem o q se preocupar.
  + **Sitemas stateless**
* **SENHA**
  + Aplicamos um algoritmo de encode. Geralmente é algum tipo de hash que gera um código a partir da palavra original. Vamos usar o **BCryptPasswordEncoder**