# Caderno de Estudos de Tokens

**Anotações dia 1 – 20/06/24: Canal Fernanda Kipper**

**Token JWT**

Tokens a grosso modo são formas de gerar autenticação pra usuários, dando determinados níveis de relevância (roles) para eles, o modelo mais popular de token é o JWT (JSON Web Token), que gera um código com três partes, a primeira sendo o cabeçalho com o tipo de criptografia e de token, a segunda sendo o corpo com os dados abertos do usuário (chamada payload) e a terceira sendo a assinatura, que carrega os dados delicados e se mantém criptografada.

O primeiro passo para fazer o JWT funcionar é importar a dependência do Spring Security no projeto (pom.xml), são a do spring-boot-starter-security e a do spring-security-test. Adicionando ela, a gente não consegue usar os endpoints da aplicação por estarem barrados, além de gerar uma tela de login e uma senha para testar ela, mas não queremos isso, queremos login através de token sendo recebido nos endpoints, verificado e enviado um token de volta.

**Anotações dia 2 – 24/06/24: Canal Fernanda Kipper**

1. Importar dependências do Spring Security
2. Tendo importado, o padrão é que os endpoints fiquem bloqueados e ele gera uma tela de login e um usuário genérico automaticamente, porém não queremos isso, e sim login via token.
3. A primeira classe que devemos criar é a de User, que representa o model dessa entidade que será armazenada no banco de dados.
4. Esse user terá um User Role, ou seja, um nível de hierarquia, para isso criamos um Enum de papeis, no nosso criamos só usuário e administrador.
5. Depois criamos uma interface UserRepository com um método não implementado para buscar login que o Java interpreta pelo nome o que deve fazer.
6. Devemos então criar um service que implementa uma classe do JPA e um método para carregar usuário por login, passando o repository que criamos anteriormente.
7. Após organizar as primeiras classes relacionadas ao login do usuário, vamos desativar o bloqueio padrão do spring security e configurar a segurança dele de acordo com o que queremos, para isso criamos uma classe de configuração.
8. Através de anotações, definimos que a classe vai ser de configuração de segurança e fazemos uma corrente de filtros para aplicar à segurança da aplicação.
9. Tendo feito as configurações de login, precisamos criar um novo endpoint para verificação de login, nesse caso criamos um controller cujo caminho é /auth de autenticação.
10. Esse controller vai fazer a autenticação do login usando uma AuthenticationManager que foi criada no SecurityConfiguration.
11. Dentro do SecurityConfiguration vamos adicionar também um método para criptografar as senhas, já que elas serão salvas assim no banco de dados.
12. No método de login do controller, vamos usar esse método de criptografia para verificar se a criptografia da senha fornecida bate com a criptografia da senha armazenada no banco de dados.