04

Validando autenticação com Hash

Transcrição

Tentamos efetuar o *login* da Ana, porém recebemos a mensagem "Usuário não encontrado". Isso aconteceu porque a senha está salva no banco de dados como código *Hash*, que é diferente de `789` passado pelo formulário. Para resolvermos essa situação usaremos a classe `BCrypt`, onde ela verificará se a senha passada pelo formulário é equivalente a senha em código *Hash* salva no banco de dados.

Pararemos o Tomcat antes de iniciarmos as modificações no código. Mudaremos a *query* para que ela se baseie apenas no e-mail passado pelo formulário. Assim, não será mais necessário mantermos o parâmetro `senha`, podemos removê-lo da *query* e também o `query.setParameter("senha", usuario.getSenha())`.

```
public Usuario procuraUsuario(Usuario usuario) {
    TypedQuery<Usuario> usuario = manager
        .createQuery("select u from Usuario u where u.email=:email", Usuario.class);
    query.setParameter("email", usuario.getEmail());
    Usuario usuarioRetornado = query.getResultList().stream().findFirst().orElse(null);
    return usuarioRetornado;
}
```

O `usuarioRetornado` será retornado apenas com base na informação do e-mail passado no formulário de *Login*. Como o objeto `usuarioRetornado` em mãos, passaremos a responsabilidade para um método onde o `BCrypt` fará a validação. Faremos uma chamada ao método `validaASenhaDoUsuarioComOHashDoBanco(usuario, usuarioRetornado)`, passando o usuário vindo do formulário e o retornado do banco de dados.

```
public Usuario procuraUsuario(Usuario usuario) {
    TypedQuery<Usuario> usuario = manager
        .createQuery("select u from Usuario u where u.email=:email", Usuario.class);
    query.setParameter("email", usuario.getEmail());
    Usuario usuarioRetornado = query.getResultList().stream().findFirst().orElse(null);

    validaASenhaDoUsuarioComOHashDoBanco(usuario, usuarioRetornado);

    return usuarioRetornado;
}
```

Como não temos o método `validaASenhaDoUsuarioComOHashDoBanco` criado, usaremos o atalho "Ctrl + 1" e selecionaremos a opção *Create Method*. Como a função do método é verificar se a senha é válida ou inválida, colocaremos o retorno no método como `boolean`.

```
public Usuario procuraUsuario(Usuario usuario) {
    TypedQuery<Usuario> usuario = manager
        .createQuery("select u from Usuario u where u.email=:email", Usuario.class);
    query.setParameter("email", usuario.getEmail());
    Usuario usuarioRetornado = query.getResultList().stream().findFirst().orElse(null);

    validaASenhaDoUsuarioComOHashDoBanco(usuario, usuarioRetornado);
```

```
    return usuarioRetornado;
}

private boolean validaASenhaDoUsuarioComOHashDoBanco(Usuario usuario, Usuario usuarioRetornado)

}
```

Se o usuário for nulo, não tem motivo de passarmos pela validação de senha. Por isso faremos um cláusula `if(usuarioRetornado == null)` , se for nulo nós retornaremos `false` .

```
// ...

private boolean validaASenhaDoUsuarioComOHashDoBanco(Usuario usuario, Usuario usuarioRetornado)

    if(usuarioRetornado == null){
        return false;
    }
}
```

Após verificar se o usuário está nulo, faremos de fato e verificação da senha vinda do formulário com a vinda do banco de dados. Usaremos o método estático `BCrypt.checkpw()` . Como o método retorna um valor `booleano` , podemos retorná-lo diretamente.

```
// ...

private boolean validaASenhaDoUsuarioComOHashDoBanco(Usuario usuario, Usuario usuarioRetornado)

    if(usuarioRetornado == null){
        return false;
    }

    return BCrypt.checkpw(usuario.getSenha(), usuarioRetornado.getSenha());
}
```

No método `procuraUsuario` , se a validação for verdadeira nós retornaremos o `usuarioRetornado` , caso contrário retornaremos `null` .

```
public Usuario procuraUsuario(Usuario usuario) {
    TypedQuery<Usuario> usuario = manager
        .createQuery("select u from Usuario u where u.email=:email", Usuario.class);
    query.setParameter("email", usuario.getEmail());
    Usuario usuarioRetornado = query.getResultList().stream().findFirst().orElse(null);

    if(validaASenhaDoUsuarioComOHashDoBanco(usuario, usuarioRetornado)){
        return usuarioRetornado;
    }

    return null;
}
```

```
private boolean validaASenhaDoUsuarioComOHashDoBanco(Usuario usuario, Usuario usuarioRetornado)

    if(usuarioRetornado == null){
        return false;
    }

    return BCrypt.checkpw(usuario.getSenha(), usuarioRetornado.getSenha());
}
```

Iniciaremos o Tomcat novamente e acessaremos a aplicação no `localhost:8080/alura-shows/`. Na página de *Login*, no campo **E-mail** colocaremos `ana@gmail.com` e no campo **Senha** colocaremos `789`. Conseguimos fazer a autenticação na aplicação.

Como teste, faremos o *Logout* na conta da Ana para entrar com uma senha errada. Novamente na página de *Login*, no campo **E-mail** colocaremos `ana@gmail.com` e no campo **Senha** colocaremos `123`. Receberemos a mensagem de "Usuário não encontrado".

Colocamos a responsabilidade da validação de senha para a classe `BCrypt` e conseguimos efetuar a autenticação utilizando senhas com código ***Hash***. Tarefa concluída.