LISTA DE EXERCÍCIO 8 – SEGURANÇA DA INFORMAÇÃO

API REST – SENHA REDUDIZA

Luan Lavandoski Guarnieri

Maria Eduarda Krutzsch

Questão 1:

Crie uma API REST para criar usuários. A API deverá receber os dados: login e senha e persistir no banco de dados, porém a senha deve ser resumida.

Resumo: a API pega o login e senha de um formulário base da web, e por meio de requisição post valida se o usuário ja esta cadastrado, valida a senha resumida para logar, se não ele cria o usuário.

senha resumida: criação de um Hash utilizando SALT

banco de dados: banco de dados em memória H2

Framework: SpringBoot

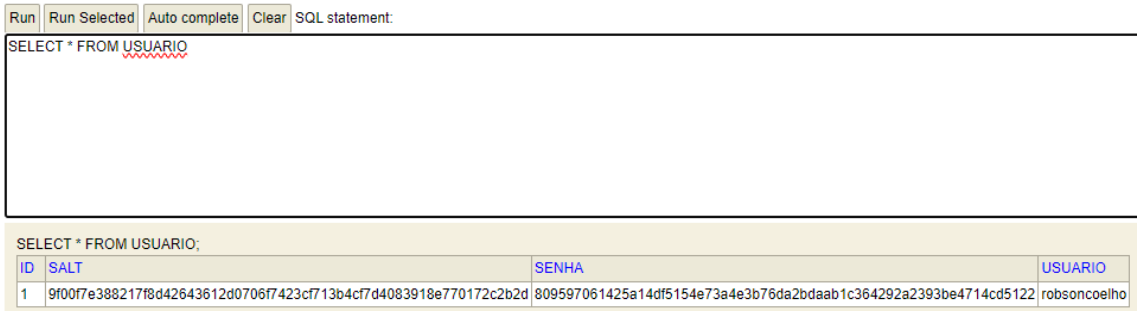
Salt: 32bytes de tamanho

Hash : algotirmo SHA-256 – tamanho de 256 bytes

usuario: ‘robsoncoelho’

senha: ‘robsinho123’

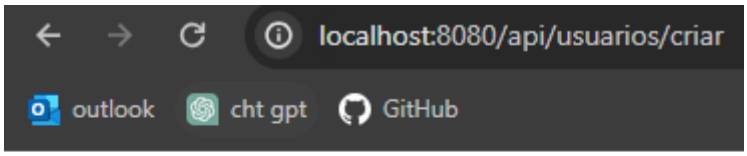
dados criados no banco: usado interface padrão do H2 para consulta



Questão 2

Cria uma API REST para realizar o login. A API deverá receber o login e a senha e conferir com o banco de dados se o login e a senha são consistentes.

Ao colocar o login: ‘robsoncoelho’ com a senha ‘robsinho123’, pela primeira vez, obtemos a mensagem:

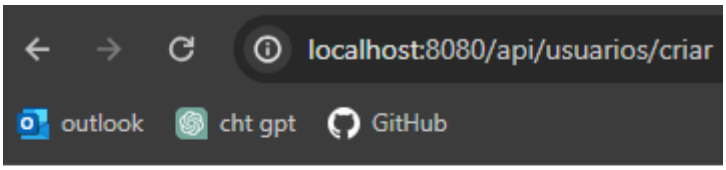


Senha do Usuário

Usuário: robsoncoelho

Usuário cadastrado com sucesso

Ao colocar o login ‘robsoncoelho’ com a senha ‘robsinho123’ nas demais vezes, obtemos a mensagem:

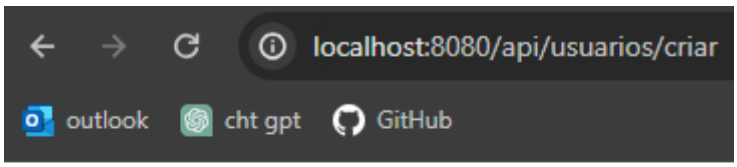


Senha do Usuário

Usuário: robsoncoelho

Usuário logado com sucesso

Ao colocar o login ‘robsoncoelho’ com outra senha, por exemplo ‘12345’, obtemos a mensagem:



Senha do Usuário

Usuário: robsoncoelho

Falha ao autenticar o usuário

Código:

Criamos a entidade usuario, o repository dela, o controller para fazer as requisições, o service para comunicar com o banco e o serviço auxiliar para a criação/ decodificação da senha

1- No controller temos dois métodos, get e post, onde o get seria o formulario que tem o login e senha e depois ele faz uma chamada post para o método criarUsuario, onde ele

vai chamar o serviço que vai fazer as validações e afins e retornar se o usuário foi criado, está autenticado ou deu falha

```java
no usages  new *
@GetMapping("/cadastro")
public String mostrarFormularioCadastro(@NotNull Model model) {
    model.addAttribute( attributeName: "usuarioPojo", new UsuarioPojo());
    return "cadastroUsuario";
}

no usages  new *
@PostMapping("/api/usuarios/criar")
public String criarUsuario(@ModelAttribute UsuarioPojo usuarioPojo, @NotNull Model model) {
    // Chamar o método desejado e receber o retorno
    String resultado = usuarioService.autenticarUsuario(usuarioPojo);
    // Adicionar o retorno ao modelo
    model.addAttribute( attributeName: "resultado", resultado);
    return "senhaResumida";
}
```

2 - No serviço de usuário, ele vai chamar o metodo autenticar usuário, que vai buscar ele no banco, caso achar chama a validação, caso contrário ele cadastra um novo

```java
1 usage  new *
public String autenticarUsuario(UsuarioPojo usuarioPojo) {
    UsuarioEntity usuario = usuarioRepository.findByUsuario(usuarioPojo.getLogin());
    return usuario != null ? validaLogin(usuarioPojo.getLogin(), usuarioPojo.getSenha()) : cadastrar(usuarioPojo.getLogin(), usuarioPojo.getSenha());
}
```

No valida login, vai chamar o autenticado, que irá buscar o hash do usuário e o salt usado na geração da senha, que por si irá chamar o serviço de senha que irá executar a geração do hash usando a senha passada pelo usuário junto com o salt do usuário recuperado em banco, que por si irá comprar se o hash é igual, caso for, ótmo, usuário autenticado, caso contrário, falha na autenticação

```java
1 usage  new *
private Boolean autenticado(String login, String senha) {
    try {
        //busca no banco o salt do usuario
        byte[] saltBanco = usuarioRepository.buscaSalt(login);
        //busca no banco o hash do usuario
        byte[] hashBanco = usuarioRepository.buscaHashSenha(login);

        //compara se a senha bate com o hash no banco
        return SenhaHashService.verificarSenha(senha, saltBanco, hashBanco);

    } catch (Exception e) {
        return false;
    }
}
```

```java
/**
 * compara senha com o hash
 *
 * @Param SENHA - senha do usuário
 * @Param SALT - o salt usado para gerar a senha
 * @Param HASHBANCO -  o hash da senha armazenado no banco
 */
1 usage  new *
public static boolean verificarSenha(String senha, byte[] salt, byte[] hashBanco) throws NoSuchAlgorithmException {
    byte[] newHash = hashSenha(senha, salt);
    return MessageDigest.isEqual(newHash, hashBanco);
}
```

3 – Caso não achar o usuário, ir´´a chamar a função cadastrar, que por si, irá chamar a cadastrarNovoUsuaio, que irá gerar o salt pelo serviço, gerar o hash do usuário usando a senha e o salt, e por fim cadastar no novo usuário em banco, gravando o hash e o salt, e retornando que o usuário foi criado com sucesso.

```
1 usage  new *
private String cadastrar(String login, String senha) {
    try {
        criarNovoUsuario(login, senha);
        return INSERIDO;
    } catch (Exception e) {
        return FALHA_CADASTRO;
    }
}

/**
 * Cria um novo usuário no banco
 */
1 usage  new *
private void criarNovoUsuario(String login, String senha) throws NoSuchAlgorithmException {
    UsuarioEntity entity = new UsuarioEntity();

    byte[] saltUsuario = SenhaHashService.gerarSalt();
    byte[] senhaUsuario = SenhaHashService.hashSenha(senha, saltUsuario);

    //gera o salt e converte em string
    entity.setSalt(saltUsuario);
    //gera a senha baseado no salt
    entity.setSenha(senhaUsuario);
    //seta o usuario
    entity.setUsuario(login);

    usuarioRepository.save(entity);
}
```

```
1 usage
private static final String HASH_ALGORITHM = "SHA-256";
// 32 bytes (256 bits)
1 usage
private static final int SALT_LENGTH = 32;

//vai gerar um salt para a senha
1 usage  new *
public static byte[] gerarSalt() {
    byte[] salt = new byte[SALT_LENGTH];
    SecureRandom secureRandom = new SecureRandom();
    secureRandom.nextBytes(salt);
    return salt;
}

//gera o hash da senha com auxilio do salt
2 usages  new *
public static byte[] hashSenha(String senha, byte[] salt) throws NoSuchAlgorithmException {
    MessageDigest digest = MessageDigest.getInstance(HASH_ALGORITHM);
    digest.reset();
    digest.update(salt);
    return digest.digest(senha.getBytes());
}
```