



{ **COTI INFORMÁTICA** }
ESCOLA DE NERDS




Java WebDeveloper - Aula 05a - Noite Seg, Qua, Sex - Noite



Tema da aula:

Projeto Herança, Criptografia MD5, Mailtrap, add, count, Filter, Stream, Listar

—  Prof Edson Belém - profedsonbelem@gmail.com (mailto:profedsonbelem@gmail.com) 🕒 Sexta, Abr 03, 2020

Criptografia MD5



O MD5 (Message-Digest algorithm 5) é uma função de dispersão criptográfica (ou função hash criptográfica) de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc., descrito na RFC 1321, e muito utilizado por softwares com protocolo ponto-a-ponto (P2P, ou Peer-to-Peer, em inglês) na verificação de integridade de arquivos e logins.

Foi desenvolvido em 1991 por Ronald Rivest para suceder ao MD4 que tinha alguns problemas de segurança. Por ser um algoritmo unidirecional, uma hash md5 não pode ser transformada novamente no texto que lhe deu origem. O método de verificação é, então, feito pela comparação das duas hash (uma da mensagem original confiável e outra da mensagem recebida). O MD5 também é usado para verificar a integridade de um arquivo através, por exemplo, do programa md5sum, que cria a hash de um arquivo. Isto pode-se tornar muito útil para downloads de arquivos grandes, para programas P2P que constroem o arquivo através de pedaços e estão sujeitos a corrupção dos mesmos. Como autenticação de login é utilizada em vários sistemas operacionais unix e em muitos sites com autenticação.

Em 2008, Ronald Rivest e outros, publicaram uma nova versão do algoritmo o MD6 com hash de tamanhos 224, 256, 384 ou 512 bits. O algoritmo MD6 iria participar do concurso para ser o novo algoritmo SHA-3, porém logo depois removeu-o do concurso por considerá-lo muito lento, anunciando que os computadores de hoje são muito lentos para usar o MD6.

O md5 é uma função criptográfica, mas não deve ser confundido com criptografia. A encriptação é uma tarefa de mão dupla que você usa sempre que precisa armazenar com segurança uma informação, mas precisa recuperá-la mais tarde através de uma chave simétrica ou privada. Já o hash, é comumente utilizado quando você necessita comparar informações.

Vulnerabilidade

Como o MD5 faz apenas uma passagem sobre os dados, se dois prefixos com o mesmo hash forem construídos, um sufixo comum pode ser adicionado a ambos para tornar uma colisão mais provável. Deste modo é possível que duas strings diferentes produzam o mesmo hash. O que não garante que a partir de uma senha codificada em hash específica consiga-se a senha original, mas permite uma possibilidade de descobrir algumas senhas a partir da comparação de um conjunto grande de hash de senhas através do método de comparação de dicionários.

Criptografia

Criptografia (em grego: kryptós, "escondido", e gráphein, "escrita") é o estudo dos princípios e técnicas pelas quais a informação pode ser transformada da sua forma original para outra ilegível, de forma que possa ser conhecida apenas por seu destinatário (detentor da "chave secreta"), o que a torna difícil de ser lida por alguém não autorizado. Assim sendo, só o receptor da mensagem pode ler a informação com facilidade. É um ramo da Matemática, parte da Criptologia. Há dois tipos de chaves criptográficas: chaves simétricas (criptografia de chave única) e chaves assimétricas (criptografia de chave pública).

Uma informação não-cifrada que é enviada de uma pessoa (ou organização) para outra é chamada de "texto claro" (plaintext). Cifragem é o processo de conversão de um texto claro para um código cifrado e decifragem é o processo contrário, de recuperar o texto original a partir de um texto cifrado. De fato, o estudo da criptografia cobre bem mais do que apenas

cifragem e decifragem. É um ramo especializado da teoria da informação com muitas contribuições de outros campos da matemática e do conhecimento, incluindo autores como Maquiavel, Sun Tzu e Karl von Clausewitz. A criptografia moderna é basicamente formada pelo estudo dos algoritmos criptográficos que podem ser implementados em computadores.

Chave (criptografia)

Uma chave criptográfica é um valor secreto que modifica um algoritmo de encriptação. A fechadura da porta da frente da sua casa tem uma série de pinos. Cada um desses pinos possui múltiplas posições possíveis. Quando alguém põe a chave na fechadura, cada um dos pinos é movido para uma posição específica. Se as posições ditadas pela chave são as que a fechadura precisa para ser aberta, ela abre, caso contrário, não.

Uma chave é um pedaço de informação que controla a operação de um algoritmo de criptografia. Na codificação, uma chave especifica a transformação do texto puro em texto cifrado, ou vice-versa, durante a decodificação. Chaves são também usadas em outros algoritmos criptográficos, tais como esquemas de assinatura digital e funções hash (também conhecidas como MAC), algumas vezes para autenticação.

Para um algoritmo bem projetado, cifrar o mesmo texto mas com uma chave diferente deverá produzir um texto cifrado totalmente diferente. Igualmente, decifrar o texto cifrado com a chave errada deverá produzir um texto aleatório ininteligível. Se a chave de deciptação for perdida, o dado cifrado praticamente não pode ser recuperado pelo mesmo algoritmo de criptografia.

As chaves usadas na criptografia de chave pública têm uma certa estrutura matemática. Por exemplo, as chaves públicas usadas no sistema RSA são o produto de dois números primos. Por isso, sistemas de chave pública requerem chaves maiores do que os sistemas simétricos para um nível equivalente de segurança. 3072bits é o tamanho de chave sugerido para sistemas baseados em fatoração e algoritmos discretos inteiros que visam ter segurança equivalente a da cifra simétrica de 128bits. A criptografia de curva elíptica (CCE) pode permitir chaves de tamanhos menores para uma segurança equivalente, mas estes algoritmos são conhecidos há pouco tempo e, pelas estimativas atuais para a dificuldade de se encontrar suas chaves, eles não devem sobreviver.

Recentemente, uma mensagem codificada usando uma chave de 109bits do algoritmo de curva elíptica foi quebrada por força bruta. A regra atual é usar uma chave de CCE com o dobro da segurança da chave simétrica para o nível desejado. Exceto para o one time pad aleatório, a segurança desses sistemas não foi provada matematicamente. Portanto, um ponto fraco teórico poderia fazer de tudo que você codificou um livro aberto. Esta é uma outra razão para se valorizar a escolha de chaves longas.

A escolha da chave

Para evitar que uma chave seja adivinhada, as chaves precisam ser geradas aleatoriamente e conterem entropia suficiente. O problema de como gerar seguramente chaves verdadeiramente aleatórias é difícil e tem sido encarado de várias formas por vários sistemas criptográficos. Existe um RFC sobre a geração de aleatoriedade (RFC 1750, Randomness Recommendations for Security [Recomendações de Aleatoriedade para Segurança]). Alguns sistemas operacionais incluem ferramentas para “coletar” entropia a partir da medição do tempo de certas operações imprevisíveis como os movimentos da cabeça do drive de disco. Para a produção de pequenas quantidades de material, um dado comum é uma boa fonte de aleatoriedade de alta qualidade.

Quando uma senha é usada como chave de codificação, os sistemas de criptografia bem projetados primeiro usam um algoritmo de derivação da chave, que adiciona um “sal” (salt) e o reduz ou o expande para o tamanho de chave desejado, por exemplo, através da redução de uma frase longa a um valor de 128bits apropriado para o uso em um bloco cifrado.

Alguns algoritmos e sistemas criptográficos

Funções de Hash criptográfico, ou message digest

- MD5
- SHA-1
- RIPEMD-160
- Tiger

Sistemas Free/Open Source

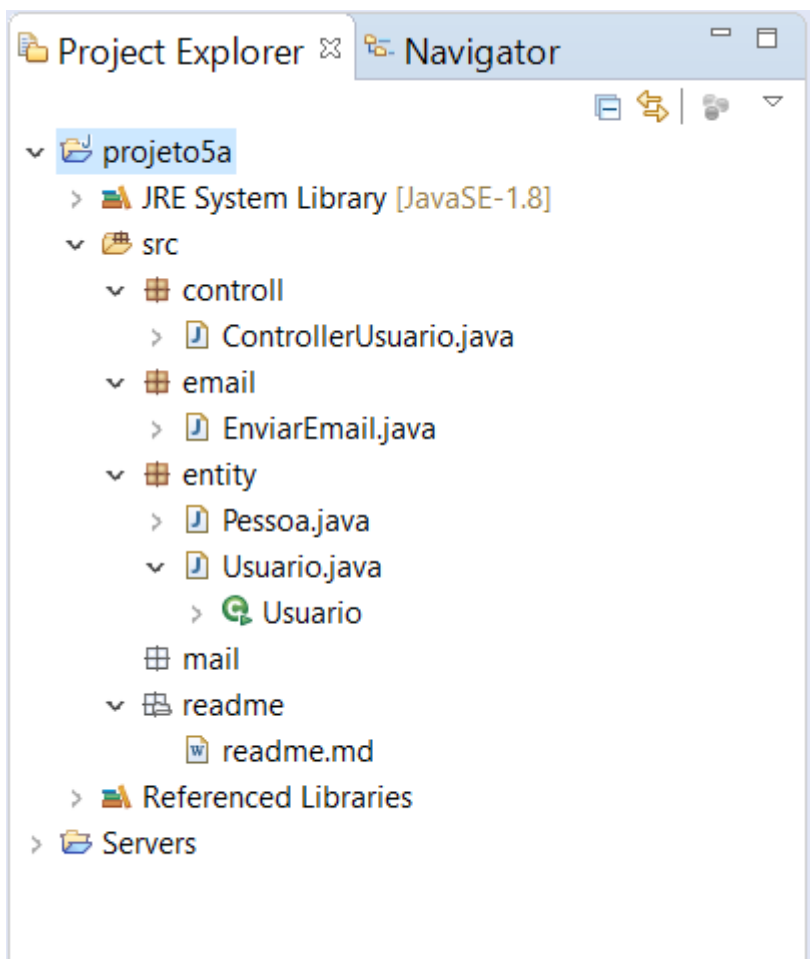
- PGP
- GPG
- SSL
- IPSec / Free S/WAN

Algoritmos assimétricos ou de chave pública

- Curvas elípticas
- Diffie-Hellman
- DSA de curvas elípticas
- El Gamal
- RSA

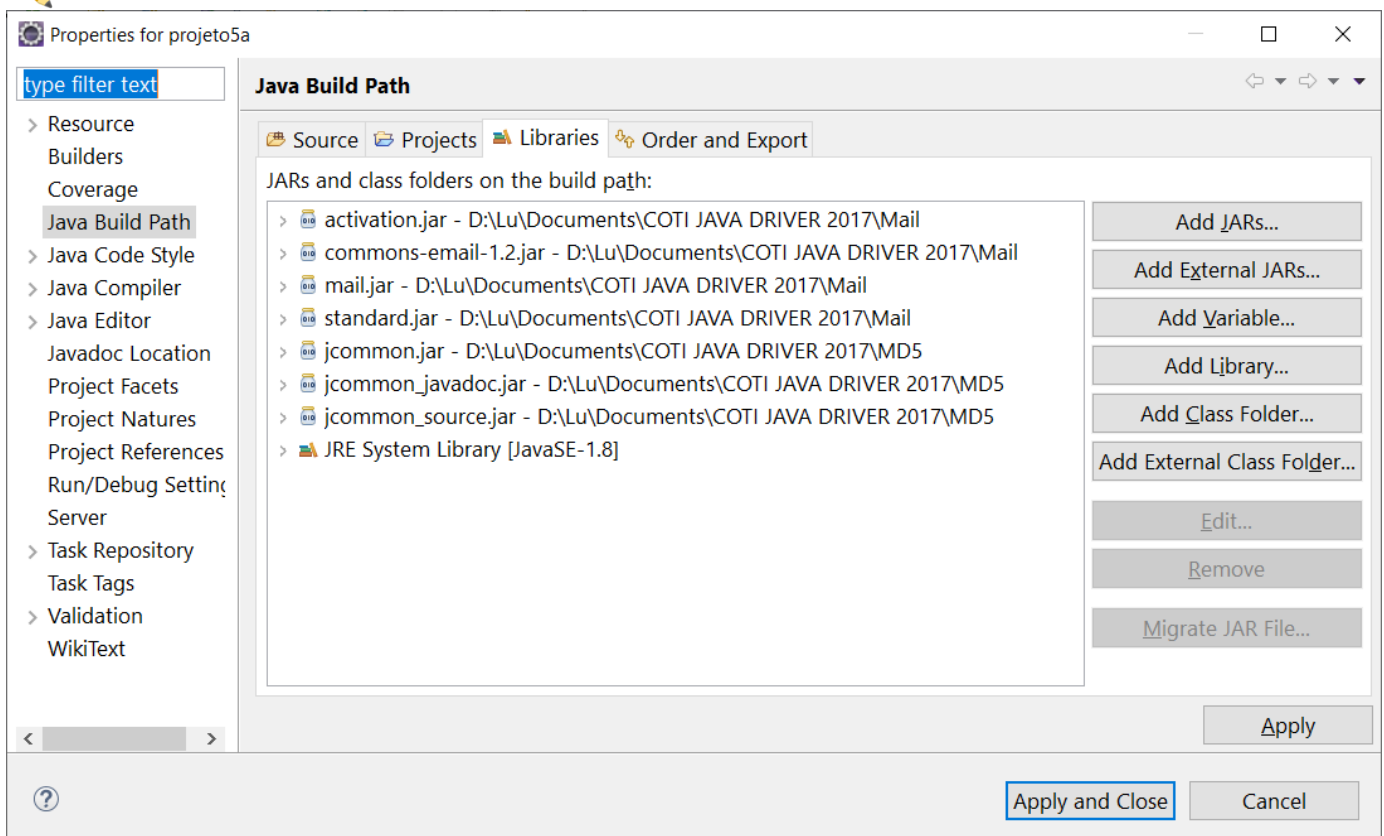


Estrutura do projeto depois de finalizado:



Bibliotecas utilizadas:

MD5 e Mail





Readme.md (<http://Readme.md>)

```

package entity;

public class Pessoa {
    // public int a = 1000;
    //alt + s
    private String nome;
    private String email;

    public Pessoa() {
    }

    //mailtrap.io

    public Pessoa(String nome, String email) {
        this.nome = nome;
        this.email = email;
    }

    @Override
    public String toString() {
        return "Pessoa [nome=" + nome + ", email=" + email + "]";
    }

    //aqui serÃ¡ uma sobrescrita
    public void display() {
        System.out.println("Classe Pessoa :" + this.getNome());
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}

=====
package entity;

import org.jcommon.encryption.SimpleMD5;

//Heranca
public class Usuario extends Pessoa{

    //public int a = 5000; //certificacao ...
    private Integer id;
    private String senha;

```

```

    public Usuario() {

    }

    public Usuario( Integer id,String nome, String email, String senha) {
        super(nome, email);
        this.id = id;
        this.senha = senha;
    }

    @Override
    public String toString() {
        return "Usuario [id=" + id + ", senha=" + senha +
        ", getNome()=" + getNome() + ", getEmail()=" + getEmail() + "]";
    }

    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }
    //Sobrescrita (Só Ocorre quando estou em heranca ou impl de interface)

    public void display() {
        System.out.println("Class Usuario :" + this);
    }

    public void gerarCriptografia() {
        SimpleMD5 md5 = new SimpleMD5(senha,
"123www.cotiinformatica.com.br@mySecret1+1=1");
        setSenha(md5.toHexString());
    }

    public static void main(String[] args) {

//        Pessoa p = new Pessoa(); // (nome,email)
//        Pessoa p2 = new Usuario(); //Classe Abstrata x interface
//        //sobrescrita ((mãe@todos iguais) = subclasse)
//        p2.display();
//        //Heranca
//        Usuario u = new Usuario();
//
//        //        Usuario x = new Pessoa();
//        //        System.out.println(p2.a);
//        //        //método sobrescreve
//        //        //atributo não sobrescreve...
//

```



```
        Usuario u = new Usuario(10,"lu","lu@gmail.com","123");
        u.gerarCriptografia();
        System.out.println(u);
    }
}
```



Pessoa.java

```
1  package entity;
2
3  public class Pessoa {
4      // public int a = 1000;
5      // alt + s
6      private String nome;
7      private String email;
8
9      public Pessoa() {
10     }
11
12     public Pessoa(String nome, String email) {
13         this.nome = nome;
14         this.email = email;
15     }
16
17     @Override
18     public String toString() {
19         return "Pessoa [nome=" + nome + ", email=" + email + "]";
20     }
21
22     // aqui será uma sobrescrita
23     public void display() {
24         System.out.println("Classe Pessoa :" + this.getNome());
25     }
26
27     public String getNome() {
28         return nome;
29     }
30
31     public void setNome(String nome) {
32         this.nome = nome;
33     }
34
35     public String getEmail() {
36         return email;
37     }
38
39     public void setEmail(String email) {
40         this.email = email;
41     }
42 }
```



Usuario.java

```
1 package entity;
2
3 import org.jcommon.encryption.SimpleMD5;
4
5 //Heranca
6 public class Usuario extends Pessoa {
7
8     // public int a = 5000; //certificacao ...
9     private Integer id;
10    private String senha;
11
12    public Usuario() {
13
14    }
15
16    public Usuario(Integer id, String nome, String email, String senha) {
17        super(nome, email);
18        this.id = id;
19        this.senha = senha;
20    }
21
22    @Override
23    public String toString() {
24        return "Usuario [id=" + id + ", senha=" + senha
25            + ", getNome()" + getNome() + ", getEmail()"
26            + getEmail() + "]";
27    }
28
29    public Integer getId() {
30        return id;
31    }
32
33    public void setId(Integer id) {
34        this.id = id;
35    }
36
37    public String getSenha() {
38        return senha;
39    }
40
41    public void setSenha(String senha) {
42        this.senha = senha;
43    }
44    // Sobrescrita (Só Ocorre quando estou em heranca ou impl
45    // de interface)
46
47    public void display() {
48        System.out.println("Class Usuario :" + this);
49    }
50
51    public void gerarCriptografia() {
52        SimpleMD5 md5 = new SimpleMD5(senha,
53            "123www.cotiinformatica.com.br@mySecret1+1=1");
54        setSenha(md5.toHexString());
```

```

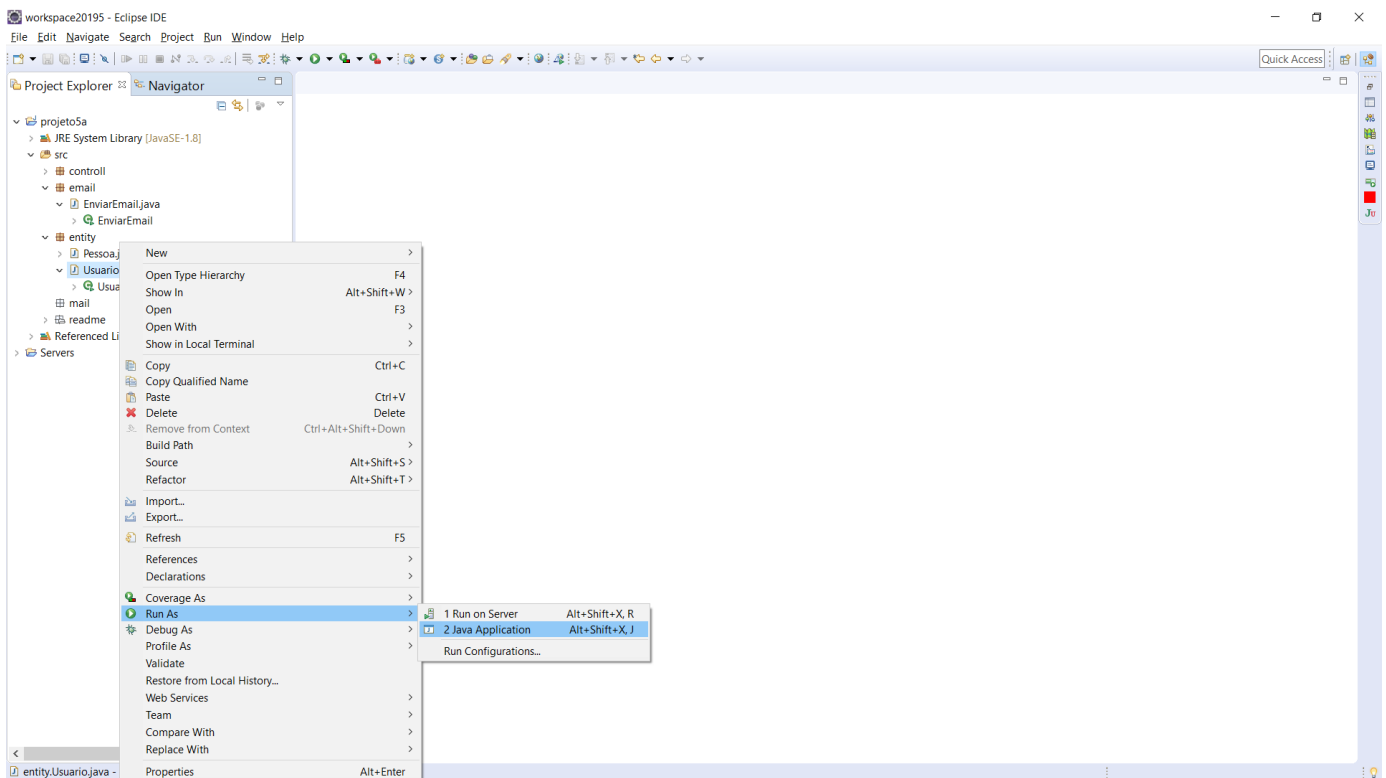
55     }
56
57     public static void main(String[] args) {
58
59         //         Pessoa p = new Pessoa(); // (nome,email)
60         //         Pessoa p2 = new Usuario(); //Classe Abstrata x interface
61         //         //sobrescrita ((métodos iguais) = subclasse)
62         //         p2.display();
63         //         //Heranca
64         //         Usuario u = new Usuario();
65         //
66         //         //         Usuario x = new Pessoa();
67         //         //         System.out.println(p2.a);
68         //         //método sobrescreve
69         //         //atributo não sobrescreve...
70         //
71         Usuario u = new Usuario(10, "lu", "lu@gmail.com", "123");
72         u.gerarCriptografia();
73         System.out.println(u);
74     }
75 }

```



Para rodar a classe:

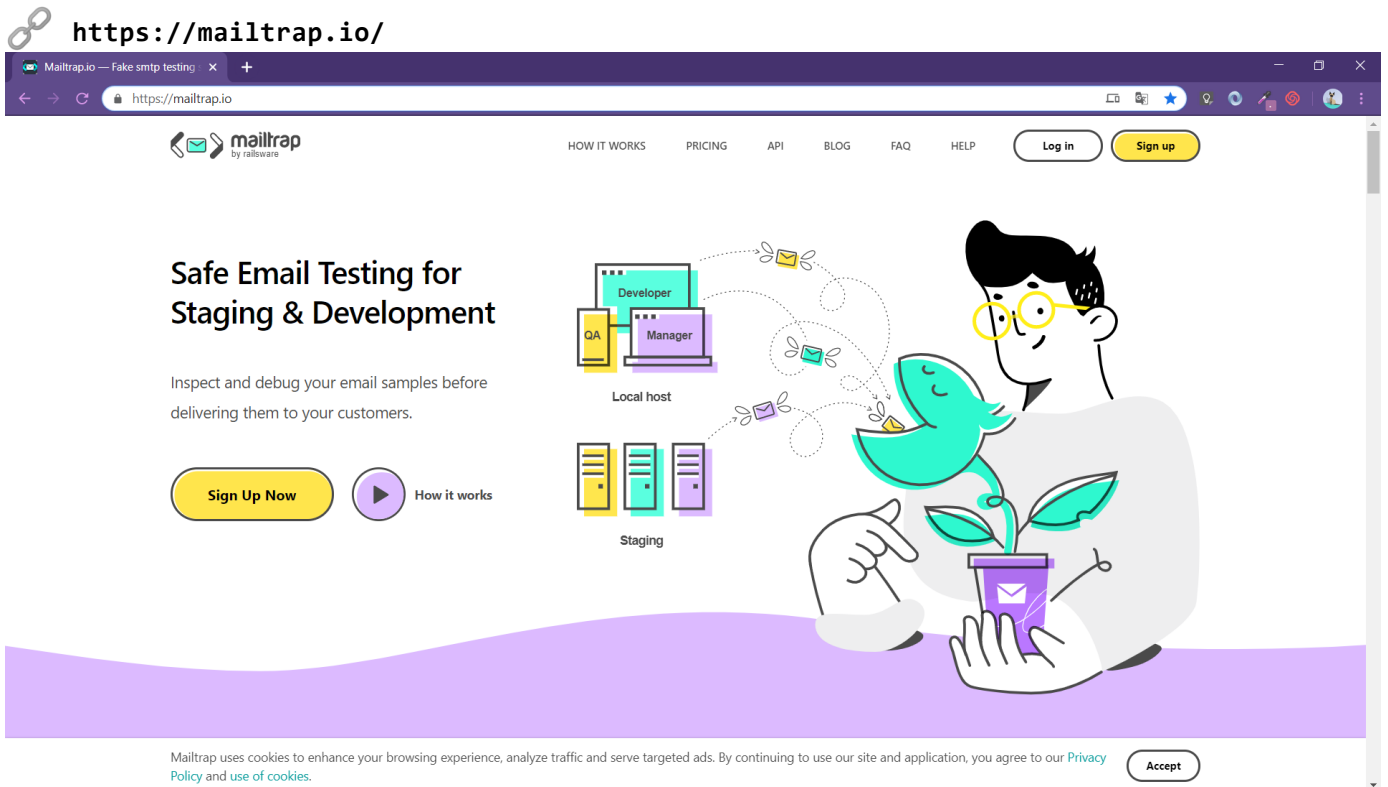
Clique na classe com o botão direito → run as → java application




Resultado no console. Já mostra a senha criptografia.

```
Properties Servers Data Source Explorer Snippets Console Progress AWS Explorer JUnit
<terminated> Usuario (2) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (6 de abr de 2020 08:33:38)
Usuario [id=10, senha=3d8a8aeef53aec242c7d5844d6cb2cbe, getNome()=lu, getEmail()=lu@gmail.com]
```

Site que iremos utilizar: Mailtrap



 Criar uma conta de email. Assim que criamos a conta recebemos um email de configuração com os dados dessa conta.

Mailtrap - Safe Email Testing x +

mailtrap.io/inboxes#

mailtrap Shared Inboxes Billing Upgrade Luciana de V Medeiros


Home New Project

My Inboxes

Inbox name Create Inbox

Inboxes	Total Sent ?	Messages	Max size	Last message	Action
Demo inbox	0	0 / 0	50	Empty	

© Copyright Mailtrap Products, Inc. All rights reserved.

 Abrindo o email com os dados

SMTP Settings Email Address Auto Forward Manual Forward Team Members

Credentials [Reset SMTP/POP3](#)

SMTP

Host: smtp.mailtrap.io
Port: 25 or 465 or 587 or 2525
Username: [REDACTED]
Password: [REDACTED]
Auth: PLAIN, LOGIN and CRAM-MD5
TLS: Optional (STARTTLS on all ports)

POP3

Host: pop3.mailtrap.io
Port: 1100 or 9950
Username: [REDACTED]
Password: [REDACTED]
Auth: USER/PASS, PLAIN, LOGIN, APOP and CRAM-MD5
TLS: Optional (STARTTLS on all ports)

Integrations

Play-Mailer

Play-Mailer is a official mailing library for Play Framework.

E-mail functionality is configured in your application's `conf/application.conf` file:

```
play.mailer {  
  host = "smtp.mailtrap.io"  
  port = 2525  
  ssl = no  
  tls = yes  
  user = "[REDACTED]"  
  password = "[REDACTED]"  
}
```




EnviarEmail.java

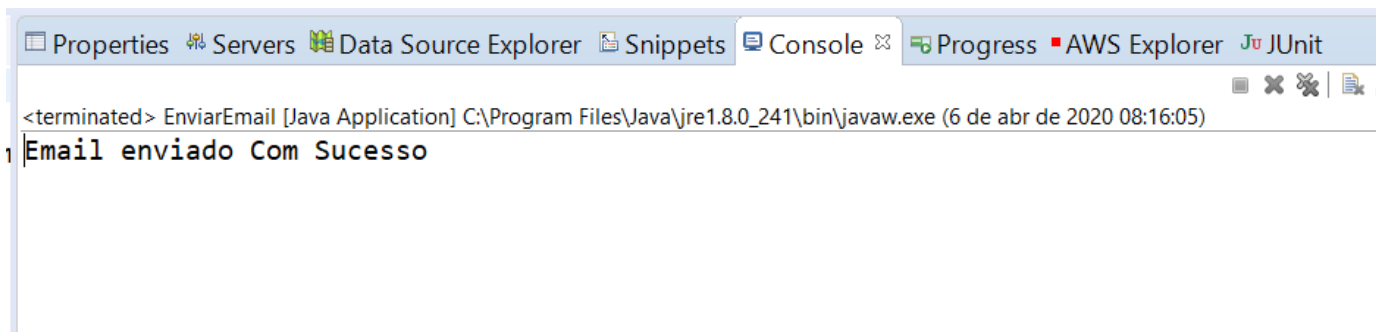
```

1  package email;
2
3  import org.apache.commons.mail.Email;
4  import org.apache.commons.mail.HtmlEmail;
5  import entity.Usuario;
6
7  public class EnviarEmail {
8      // adicionar os jar do mail
9      // loguei ou criei uma conta no mailtrap
10     /*
11      * play.mailer { host = "smtp.mailtrap.io" port = 2525 ssl = no tls
12      * = yes user =
13      * "3d80cc6ddf5564g713b67" password = "7c3rgy6783cd2af7940" }
14      */
15
16     //LEMBRE-SE DE USAR A SUA CONTA
17     public static final String HOSTNAME = "smtp.mailtrap.io";
18     public static final Integer PORT = 2525;
19     public static final Boolean SSL = false;
20     public static final Boolean TLS = true;
21     public static final String USERNAME = "3d80cc6ddf5564g713b67";
22     public static final String PASSWORD = "7c3rgy6783cd2af7940";
23
24     // Estou me Conectando
25     public static Email conectaEmail() throws Exception {
26         HtmlEmail mail = new HtmlEmail();
27         mail.setHostName(HOSTNAME);
28         mail.setSmtpport(PORT);
29         mail.setAuthentication(USERNAME, PASSWORD);
30         mail.setTLS(TLS);
31         mail.setSSL(SSL);
32         return mail;
33     }
34
35     public static String enviarEmail(Usuario u) throws Throwable {
36         HtmlEmail email = new HtmlEmail();
37         email = (HtmlEmail) conectaEmail();
38         email.setFrom("cotiead@gmail.com");
39         email.setSubject("Seja Bem Vindo ao Ambiente Coti EAD");
40         email.addTo(u.getEmail());
41         email.setMsg("<h2>::: "
42             + "Entre no site www.cotiead.com.br</h2>"
43             + "<hr/><br/> seja bem Vindo ao Coti = <b>"
44             + u.getNome() + "</b>");
45
46         Thread.sleep(1000); // da um tempo de 1 segundo
47         email.send(); // envia o email (mailtrap)
48         return "Email enviado Com Sucesso";
49     }
50
51     public static void main(String[] args) {
52         Usuario u = new Usuario(100, "belem",
53             "profedsonbelem@gmail.com", "123");
54         try {

```

```
55         String resp = EnviarEmail.enviarEmail(u);
56         System.out.println(resp);
57     } catch (Throwable tx) {
58         tx.printStackTrace();
59     }
60 }
61 }
```

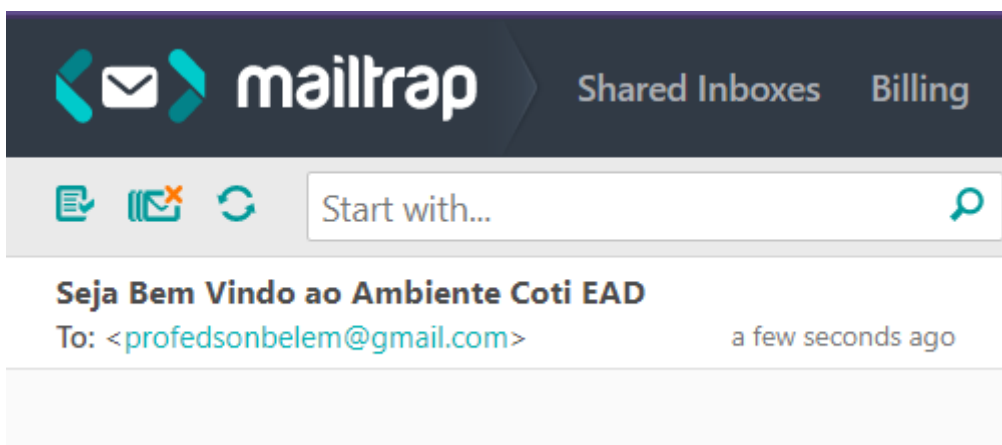
 Resultado no console:



The screenshot shows an IDE with several tabs: Properties, Servers, Data Source Explorer, Snippets, Console, Progress, AWS Explorer, and JUnit. The Console tab is active, displaying the output of a Java application named 'EnviarEmail'. The output shows the message 'Email enviado Com Sucesso'.

```
<terminated> EnviarEmail [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (6 de abr de 2020 08:16:05)
Email enviado Com Sucesso
```

 No Mailtrap



 Abrindo o email

Seja Bem Vindo ao Ambiente Coti EAD

From: cotiead@gmail.com <cotiead@gmail.com>
To: profedsonbelem@gmail.com <profedsonbelem@gmail.com>

[Show Info](#)

HTML

HTML Source

Text

Raw

Analysis

Check HTML

SMTP info

::: Entre no site www.cotiead.com.br

seja bem Vindo ao Coti = **belem**



ControllerUsuario.java

```
1 package controll;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import entity.Usuario;
7
8 public class ControllerUsuario {
9     //Mockado
10     public static List<Usuario> usuarios = new ArrayList<Usuario>();
11     static {
12         Usuario u1 = new Usuario(10, "lu", "lu@gmail.com", "123");
13         u1.gerarCriptografia();
14         Usuario u2 = new Usuario(20, "marcos", "marcos@gmail.com", "123");
15         u2.gerarCriptografia();
16         usuarios.add(u1);
17         usuarios.add(u2);
18     }
19
20     public static void adicionar(Usuario u) {
21         u.gerarCriptografia();
22         usuarios.add(u);
23     }
24
25     public static int count() {
26         return usuarios.size();
27     }
28
29     public static List<Usuario> findAll() {
30         return usuarios;
31     }
32
33     //stream
34     public static Usuario findById(Integer id) {
35         Usuario resposta = usuarios.stream().filter(a -> {
36             return a.getId().equals(id);
37         }).findAny().orElse(null);
38         return resposta;
39     }
40
41     public static void main(String[] args) {
42         ControllerUsuario.findAll().stream()
43             .forEach(x -> System.out.println(x.getNome() + ","
44             + x.getEmail()));
45
46         System.out.println(count());
47
48         Usuario y = new Usuario(30, "belem", "bel@gmail.com", "123");
49         ControllerUsuario.adicionar(y);
50
51         System.out.println("H==>" + findAll());
52         System.out.println(count());
53
54         Usuario resp = findById(3);
```

```
55         if (resp == null) {
56             System.out.println("Nao Encontrado o Usuario");
57         } else {
58             System.out.println("Encontrado:" + resp);
59         }
60     }
61 }
```

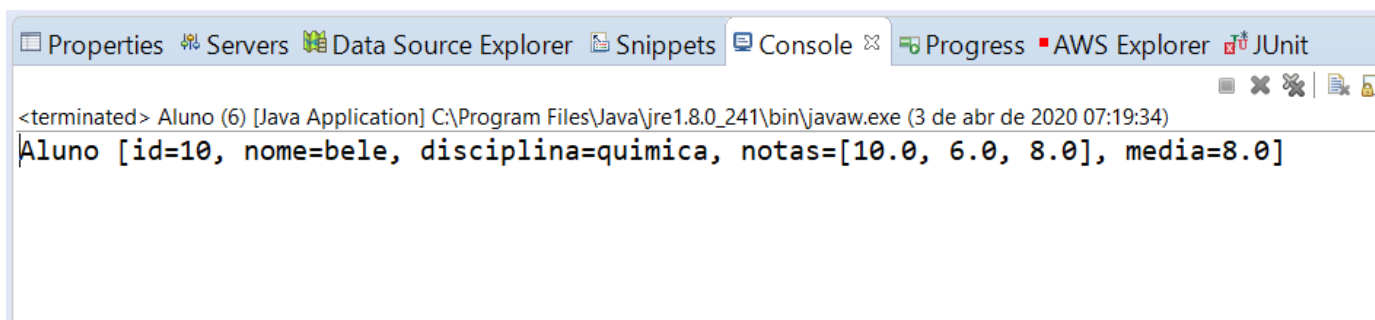


Para rodar a classe:

Clique na classe com o botão direito ➡ run as ➡ java application




Resultado no console.



The screenshot shows an IDE interface with a console window. The console title bar includes tabs for Properties, Servers, Data Source Explorer, Snippets, Console, Progress, AWS Explorer, and JUnit. The console output displays the following text:

```
<terminated> Aluno (6) [Java Application] C:\Program Files\Java\jre1.8.0_241\bin\javaw.exe (3 de abr de 2020 07:19:34)
Aluno [id=10, nome=bele, disciplina=quimica, notas=[10.0, 6.0, 8.0], media=8.0]
```

—  Coti Informática <https://www.cotiinformatica.com.br> (<https://www.cotiinformatica.com.br>)