Método que nos permite saber que interes se tiene que aplicar de acuerdo a los días que solicite la apoliza

```
public Double obtenerInteres(int tiempo) {

    if(tiempo>=30 && tiempo <=59) {
    return 5.50 ;
    }else
        if(tiempo>=60 &&  tiempo <=89) {
        return 5.75;
        }else
            if(tiempo>=90 &&  tiempo <=179) {
                return 6.25;
                }else
                    if(tiempo>=180 &&  tiempo <=269) {
                        return 7.00;
                        }else
                            if(tiempo>=270 &&  tiempo <=359) {
                                return 7.50;
                                }else
                                    if(tiempo>=360) {
                                        return 8.50;
                                        }
    return null;


}
```

En este método no sirve para simular la polia dando de entrada el monto y los días que solicita la póliza

```
 */
public String tablaAmor(){
        interes= obtenerInteres(dias);
       total =((monto*interes)*dias)/100;
            System.out.println("monto inicial "+monto);
            System.out.println("interes es " + interes);
            System.out.println("valor total " + total);

      return null;
}
```

Métodos que permite verificar la sesión de un usuario y del amig

```java
    /**
     * metodo para verificar la sesion del cliente
     */
    public void verificarSession() {
        try {
            FacesContext context = FacesContext.getCurrentInstance();
            Cliente cliente = (Cliente) context.getExternalContext().getSessionMap().get("cliente");

            if (cliente == null) {
                context.getExternalContext().redirect("InicioClientes.xhtml");
            }

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

    }
```

```java
     * metodo para verificar la sesion de los empleados
     */

    public void verificarSessionAdmin(){
            try {
                FacesContext context = FacesContext.getCurrentInstance();
                Empleado  empleado= (Empleado) context.getExternalContext().getSessionMap().get("empleado");
                if (empleado == null) {
                    context.getExternalContext().redirect("InicioUsuarios.xhtml");
                }

            } catch (Exception e) {
                System.out.println(e.getMessage());

            }
    }
}
```

Método que permite mostrar la cuenta mediante la cedula primero validadndo si esta correcta la cedula y luego se busca la cuenta de acuerdo a a cedula

```java
 */
public String numCuenta() {

    try {
        boolean c;
        c = clienteON.validadorDeCedula(cedulaCliente);
        if (c) {
            Cliente usuarioRegistrado = clienteON.buscarCliente(cedulaCliente);
            if (usuarioRegistrado != null) {
                CuentaDeAhorro cuen = cuentaON.buscarCuentaDeAhorroCliente(usuarioRegistrado.getCedula());
                return cuen.getNumeroCuentaDeAhorro();
            }
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        // e.printStackTrace();
    }
    return null;

}
```

Método que permite validar que el monto a retirar no sea mayor al que tiene el usuario en la cuenta que tiene un usuario en su cuenta

```
 */
public String valMonto() {
    try {
        if (cedulaCliente != null) {
            Cliente usuarioRegistrado = clienteON.buscarCliente(cedulaCliente);
            if (usuarioRegistrado != null) {
                Cuenta cl = cuentaON.buscarCuentaDeAhorroCliente(cedulaCliente);
                String l = String.valueOf(cl.getSaldoCuentaDeAhorro());
                if (tipoTransaccion.equalsIgnoreCase("retiro") && monto == null) {
                    return l;
                } else if (tipoTransaccion.equalsIgnoreCase("retiro") && cl.getSaldoCuentaDeAhorro() < monto) {
                    String ms = "La cuenta no cuenta con el saldo suficiente, Su saldo es: "
                            + cl.getSaldoCuentaDeAhorro();
                    return ms;

                } else {
                    return l;
                }

            }
        }

    } catch (Exception e) {
        // TODO: handle exception
    }
```

Métodos que permite enviar un correo mediante smtp

```
31
32
33      public void enviarCorreo(String destinatario, String asunto, String cuerpo) {
34          Properties propiedad = new Properties();
35          propiedad.setProperty("mail.smtp.host", "smtp.gmail.com");
36          propiedad.setProperty("mail.smtp.starttls.enable", "true");
37          propiedad.setProperty("mail.smtp.port", "587");
38
39          Session sesion = Session.getDefaultInstance(propiedad);
40          String correoEnvia = "banconet123@gmail.com";
41          String contrasena = "b_net123";
42
43          MimeMessage mail = new MimeMessage(sesion);
44          try {
45              mail.setFrom("BANCONET <" + correoEnvia + ">");
46              mail.addRecipient(Message.RecipientType.TO, new InternetAddress(destinatario));
47              mail.setSubject(asunto);
48              mail.setText(cuerpo);
49
50              Transport transportar = sesion.getTransport("smtp");
51              transportar.connect(correoEnvia, contrasena);
52              transportar.sendMessage(mail, mail.getRecipients(Message.RecipientType.TO));
53          } catch (AddressException ex) {
54              System.out.println(ex.getMessage());
55          } catch (MessagingException ex) {
56              System.out.println(ex.getMessage());
57          }
58      }
59
```

Método que permite generar un numero de cuenta randomicamente lista la cuenta y de lo que tiene el tamaño de la cuenta le suma 1 al resultado

```
    public String generarNumeroDeCuenta() {
        int numeroInicio =100;
        List<CuentaDeAhorro> listaCuentas = listaCuentaDeAhorros();
        int numero = listaCuentas.size() + 1;
        String resultado = String.format("%08d", numero);
        String resultadoFinal = String.valueOf(numeroInicio) + resultado;
        return resultadoFinal;
    }

    /**
```

Método que permite que permite generar un usuario de acuerdo a su nombre y apellido

```
 */
public String getUsuario(String cedula, String nombre, String apellido) {
    System.out.println(cedula);
    System.out.println(nombre);
    System.out.println(apellido);
    String ud = cedula.substring(cedula.length() - 1);
    String pln = nombre.substring(0, 1);
    int it = 0;
    for (int i = 0; i < apellido.length(); i++) {
        if (apellido.charAt(i) == 32) {
            it = i;
        }
    }
    String a = "";
    if (it == 0) {
        a = apellido.substring(0, apellido.length());
    } else {
        a = apellido.substring(0, it);
    }
    return pln.toLowerCase() + a.toLowerCase() + ud;
}
```

Método para obtener una contraseña randomicamente

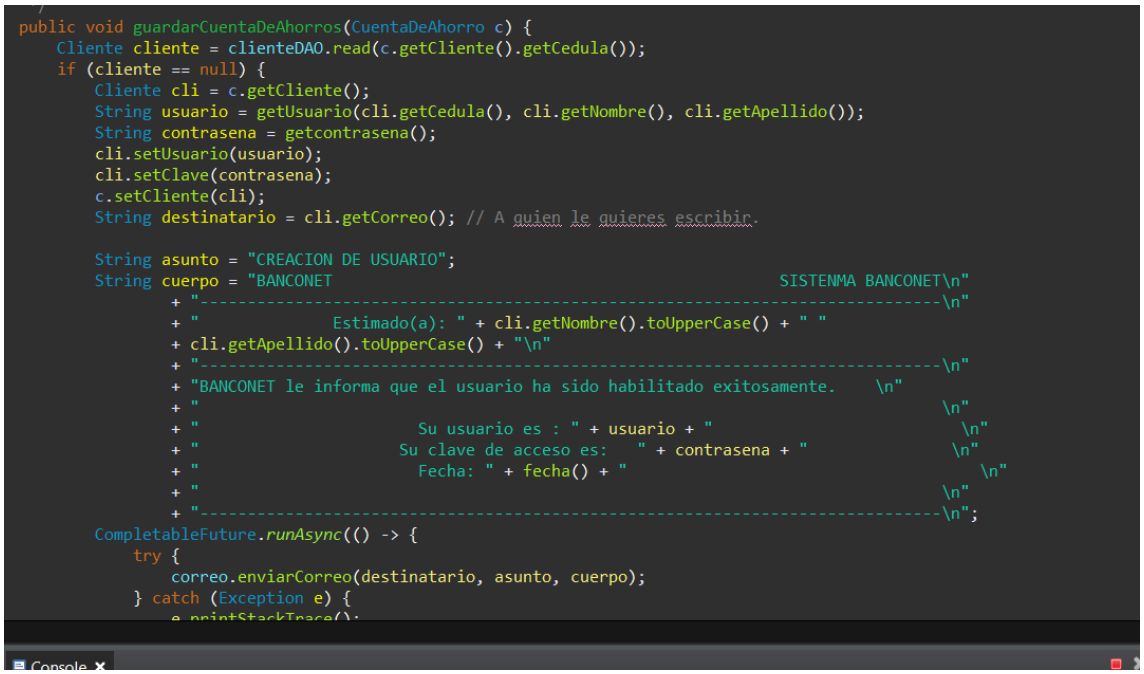
```
     * @return contrasena aleatoria
     */
    public String getcontrasena() {
        String simbolos = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefjhijklmnopqrstuvwxyz0123456789";

        int tam = simbolos.length() - 1;
        System.out.println(tam);
        String clave = "";
        for (int i = 0; i < 10; i++) {
            int v = (int) Math.floor(Math.random() * tam + 1);
            clave += simbolos.charAt(v);
        }

        return clave;
    }
```

Ç

Método que permite el registro de la cuenta lo primero que hace es buscar el usuario y si es null procede a guardar el cliente y luego se procede escribir el cuerpo, el asunto del mensaje mostrando el usuario y la contraseña al correo



```
public void guardarCuentaDeAhorros(CuentaDeAhorro c) {
    Cliente cliente = clienteDAO.read(c.getCliente().getCedula());
    if (cliente == null) {
        Cliente cli = c.getCliente();
        String usuario = getUsuario(cli.getCedula(), cli.getNombre(), cli.getApellido());
        String contrasena = getcontrasena();
        cli.setUsuario(usuario);
        cli.setClave(contrasena);
        c.setCliente(cli);
        String destinatario = cli.getCorreo(); // A quien le quieres escribir.

        String asunto = "CREACION DE USUARIO";
        String cuerpo = "BANCONET                                                    SISTENMA BANCONET\n"
                + "-------------------------------------------------------------------------------\n"
                + "             Estimado(a): " + cli.getNombre().toUpperCase() + " "
                + cli.getApellido().toUpperCase() + "\n"
                + "-------------------------------------------------------------------------------\n"
                + "BANCONET le informa que el usuario ha sido habilitado exitosamente.    \n"
                + "                                                                          \n"
                + "                      Su usuario es : " + usuario + "                       \n"
                + "                    Su clave de acceso es:    " + contrasena + "            \n"
                + "                      Fecha: " + fecha() + "                                  \n"
                + "                                                                          \n"
                + "-------------------------------------------------------------------------------\n";
        CompletableFuture.runAsync(() -> {
            try {
                correo.enviarCorreo(destinatario, asunto, cuerpo);
            } catch (Exception e) {
                e.printStackTrace();
```

Console

Métodos que permite guardar la sesión del cliente y manda un correo diciento si ha ingresado correctamente, incorrectamente, usuario bloqueado

```
public void guardarSesion(SesionCliente sesionCliente,int fallidos) {
    Cliente cli = sesionCliente.getCliente();
    String destinatario = cli.getCorreo();

    if(fallidos==3) {
        String asunto = "USUARIO BLOQUEADO";
        String cuerpo = "BANCONET  \n"
                + "-------------------------------------------------------------------------------\n"
                + "             Estimado(a): " + cli.getNombre().toUpperCase() + " "
                + cli.getApellido().toUpperCase() + "\n"
                + "-------------------------------------------------------------------------------\n"
                + "BANCONET le informa que el acceso a su cuenta ha sido bloqueada por favor \n  "
                + "comunicarse con el banco   \n"
                +"MOTIVO DE BLOQUEO   \n"
                + "  INTENTOS DE INGRESAR A LA CUENTA      "+ fallidos +"\n"
                + "   Fecha: " + obtenerFecha(sesionCliente.getFechaSesion())
                + "                                    \n"
                + "                                                                          \n"
                + "-------------------------------------------------------------------------------\n";
        CompletableFuture.runAsync(() -> {
            try {
                correo.enviarCorreo(destinatario, asunto, cuerpo);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }else
```

```
    });
}else
if (sesionCliente.getEstado().equalsIgnoreCase("Incorrecto")) {
    // A quien le quieres escribir.

    String asunto = "INICIO DE SESION FALLIDA";
    String cuerpo = "BANCONET  \n"
            + "-------------------------------------------------------------------------------\n"
            + "              Estimado(a): " + cli.getNombre().toUpperCase() + " "
            + cli.getApellido().toUpperCase() + "\n"
            + "-------------------------------------------------------------------------------\n"
            + "BANCONET le informa que el acceso a su cuenta ha sido fallida.    \n"
            + "  INTENTOS FALLIDOS      "+fallidos
            + "   Fecha: " + obtenerFecha(sesionCliente.getFechaSesion())
            + "                                    \n"
            + "                                                                    \n"
            + "-------------------------------------------------------------------------------\n";
    CompletableFuture.runAsync(() -> {
        try {
            correo.enviarCorreo(destinatario, asunto, cuerpo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    });

} else {
```

```
    // A quien le quieres escribir.

    String asunto = "INICIO DE SESION CORRECTA";
    String cuerpo = "BANCONET \n"
            + "-------------------------------------------------------------------------------\n"
            + "              Estimado(a): " + cli.getNombre().toUpperCase() + " "
            + cli.getApellido().toUpperCase() + "\n"
            + "-------------------------------------------------------------------------------\n"
            + "BANCONET le informa que el acceso a su cuenta ha sido correcta.    \n"
            + "                       Fecha: " + obtenerFecha(sesionCliente.getFechaSesion())
            + "                                    \n"
            + "                                                                    \n"
            + "-------------------------------------------------------------------------------\n";

    CompletableFuture.runAsync(() -> {
        try {
            correo.enviarCorreo(destinatario, asunto, cuerpo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    });
}
```

Método que permite validar al usuario de acuerdo al rol y dirigir a la pagina

```
public String validarUsuario() {
    Empleado emp;
    try {
        emp = empleadoON.usuario(usuario, contrasena);
        System.out.println("******  USUARIO ****" + emp.getNombre());
        empleado = emp;
        if (emp != null && emp.getRol().equalsIgnoreCase("Cajero")) {
            try {
                addMessage("OK", "Ingreso");


                FacesContext.getCurrentInstance().getExternalContext().getSessionMap().put("empleado", emp);
                //FacesContext contex2 = FacesContext.getCurrentInstance();
                FacesContext contex = FacesContext.getCurrentInstance();

                contex.getExternalContext().redirect("PaginaCajero.xhtml");
            } catch (Exception e) {
            }
        } else if (emp != null && emp.getRol().equalsIgnoreCase("AsistenteCaptacion")) {
            try {
                FacesContext contex = FacesContext.getCurrentInstance();
                FacesContext.getCurrentInstance().getExternalContext().getSessionMap().put("empleado", emp);
                contex.getExternalContext().redirect("PaginaJefePoliza.xhtml");
            } catch (Exception e) {
            }
        } else if (emp != null && emp.getRol().equalsIgnoreCase("Admin")) {
            try {
                FacesContext contex = FacesContext.getCurrentInstance();
                FacesContext.getCurrentInstance().getExternalContext().getSessionMap().put("empleado", emp);
                contex.getExternalContext().redirect("Admin.xhtml");
```

Método para crear al usuario llamado desde el vean

```
public String crearCuenta() {
    try {
        cuentaDeAhorro.setNumeroCuentaDeAhorro(numeroCuenta);
        cuentaDeAhorro.setFechaDeRegistro(new Date());
        cuentaDeAhorro.setTipoCuenta(tipoCuenta);
        cuentaDeAhorro.setCliente(cliente);
        System.out.println("saldo es "+saldoCuenta+ "tipo de cuenta"+tipoCuenta);
        cuentaDeAhorro.setSaldoCuentaDeAhorro(Double.parseDouble(saldoCuenta));
        gestionCuenta.guardarCuentaDeAhorros(cuentaDeAhorro);
        Transaccion transaccion = new Transaccion();
        transaccion.setFecha(new Date());
        transaccion.setMonto(cuentaDeAhorro.getSaldoCuentaDeAhorro());
        transaccion.setTipo("deposito");
        transaccion.setCliente(cliente);
        transaccion.setSaldoCuenta(cuentaDeAhorro.getSaldoCuentaDeAhorro());
        gestionTransaccion.guardarTransaccion(transaccion);
        addMessage("Confirmacion", "Cliente Guardado");
        cliente = new Cliente();
        try {
            FacesContext contex = FacesContext.getCurrentInstance();
            contex.getExternalContext().redirect("CrearCliente.xhtml");
        } catch (Exception t) {

        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return null;
```