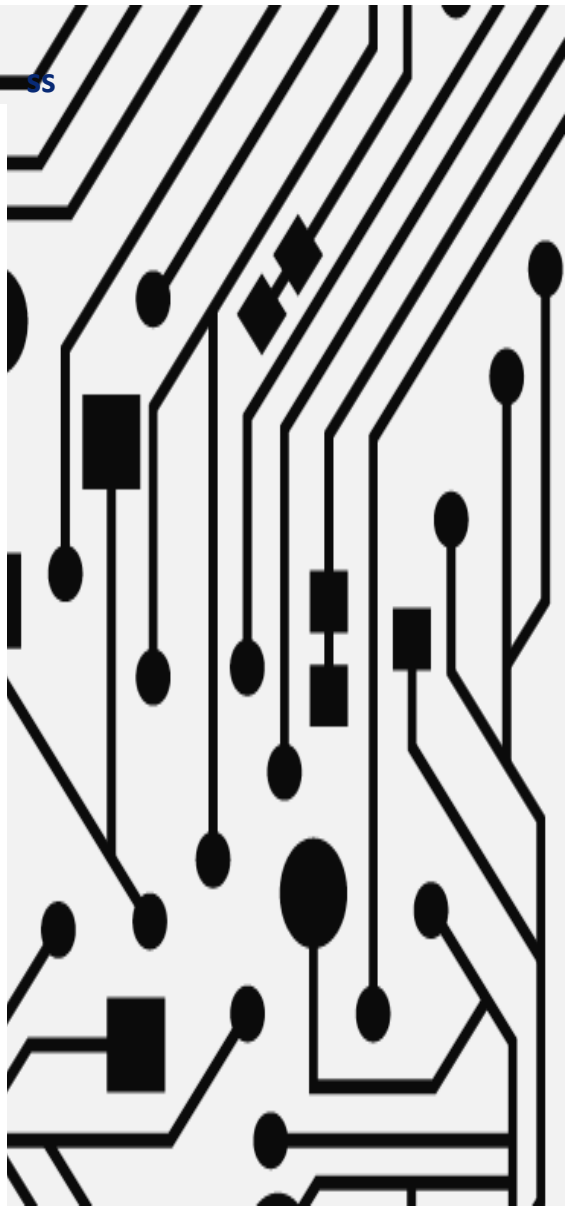
SS

Manual del programador



DolphinSoft

Creado por: Grupo Dolphin

Scrum Master: Miguel Ángel Ayala Bejarano

Scrum Team:

Gabriel Antonio Castillo Alegría

Reynaldo Daniel Panameño Romero

Carlos Humberto Posada Gaitán

Indice

Contenido

1.Introducción: 4
2.Objetivos: 5
3. Requerimientos..... 6
4. Diagrama DDC yDCU..... 7
5.Diagrama Entidad-Relación 11
6. Descripción de las tablas.....12
- Tabla usuario 12
-Tabla roles 12
-Tabla empleados.....12
-Tabla InfoPago 13
-Tabla Pagoempleados 14
7. Descripcion de las variables usadas en el sistema 15
-Funciones y clases 18
8.Diccionario de datos. 44
Tabla usuario 44
Tabla roles 45
Tabla empleados 45
Tabla pagoempleados.....46
Tabla infopago 46
Detalles especiales de ejecución 47
Prerequisitos:.....47
Usos especificos de recursos: 47
Librerias para generar los reportes: 47
4 Librerias para encryptar y desencriptar las contraseñas y hacer uso de la clase seguridad.java: 48

Librería para conectar la base de datos:.....48

Descripcion de interfaces con otros sistemas o aplicaciones 48

Bitacoras de cambios dentro de los mismos. 49

1.Introducción:

El presente documento describe los aspectos técnicos informáticos del sistema de información. El documento familiariza al personal técnico especializado encargado de las actividades de mantenimiento, revisión, solución de problemas, instalación y configuración del sistema.

Este manual técnico hace referencia a la información necesaria con el fin de orientar al personal encargado, planteamiento análisis programación e instalación del sistema. Es de notar que la redacción propia de este manual está orientada a personal con conocimientos en sistemas y tecnologías de información, conocimientos de programación, administración de bases de datos, responsables del mantenimiento e instalación del sistema.

2.Objetivos:

Instruir el uso adecuado del Sistema de Información, para el acceso oportuno y adecuado en la instalación del mismo, mostrando los pasos a seguir en el proceso de instalación, así como la descripción de los archivos relevantes del sistema los cuales nos orienten en la configuración y soporte del mismo.

De igual manera se busca ampliar conocimientos acerca de la estructura y funcionamiento del sistema con el interés de difundir estos conocimientos a otros usuarios que se interesen en el manejo de este sistema.

3. Requerimientos

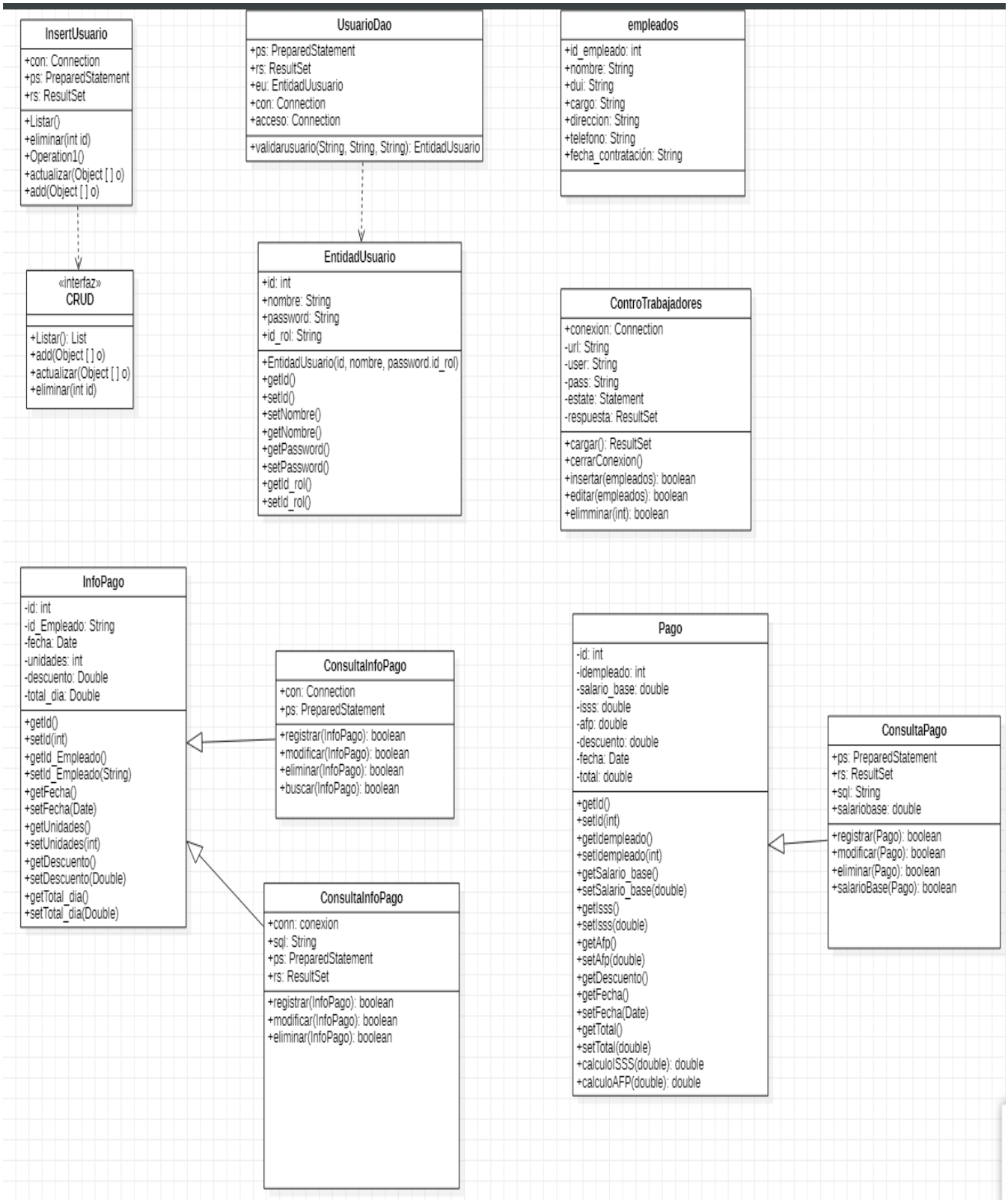
El sistema pude ser instalado en cualquier sistema operativo que cumpla con los siguientes requerimientos:

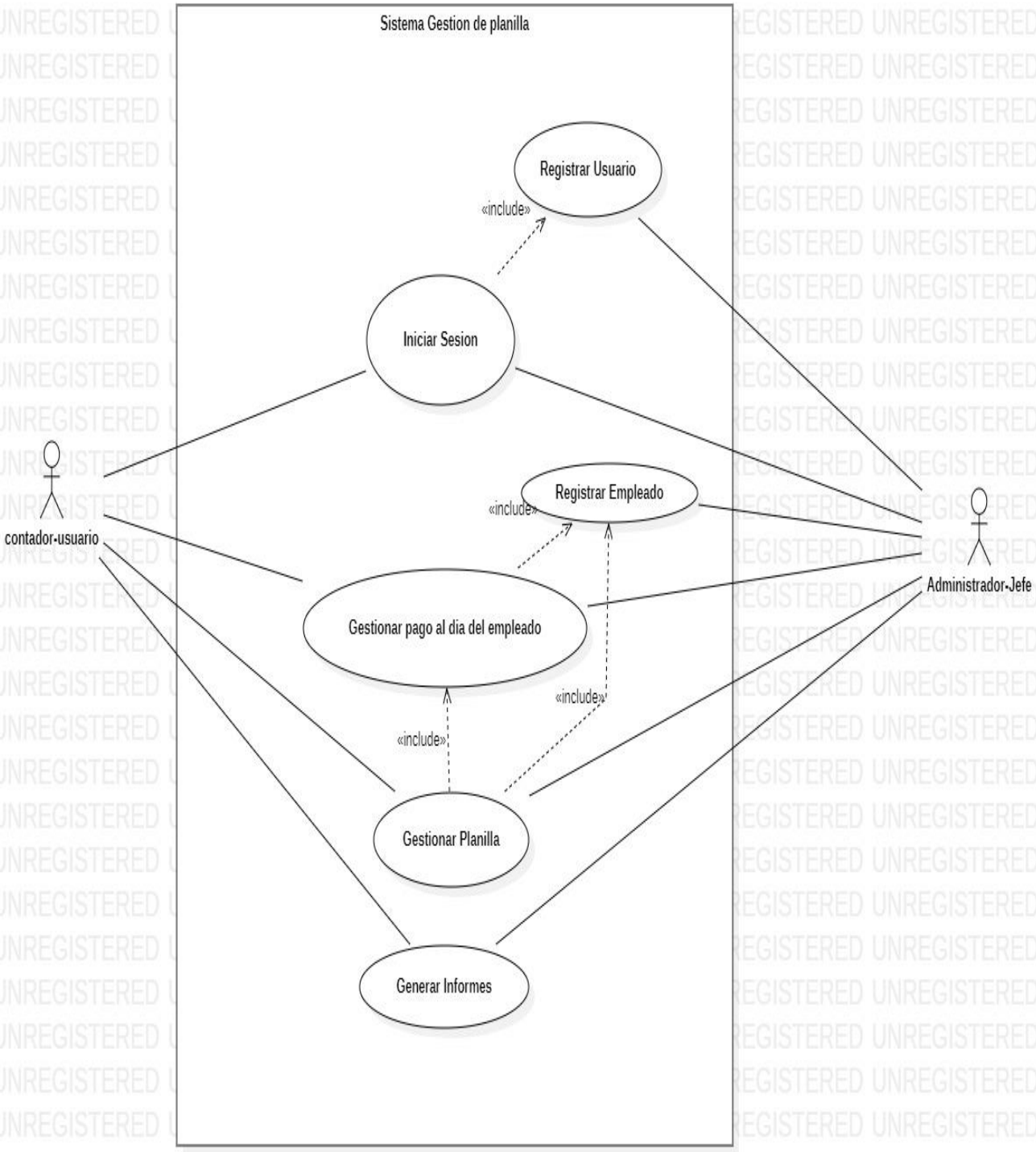
-Netbeans Apache 12.2

-Servidor web Apache 2.0 o superior.

-Sistema operativo Windows o Linux.

4. Diagrama DDC yDCU




Sistema Gestion de planilla
Registrar Usuario
«include»
Iniciar Sesion
Registrar Empleado
«include»
Gestionar pago al dia del empleado
«include»
«include»
Gestionar Planilla
Generar Informes
contador-usuario
Administrador-Jefe

CASO DE USO:	REGISTRAR USUARIO
ACTORES:	**Administrador**
PRECONDICION:	**Es necesario validar usuarios repetidos**
DESCRIPCIÓN:	**El administrador registrar a los usuarios al sistema, para que estos pueden ingresar en el login(iniciar sesión)**

CASO DE USO:	INICIAR SESIÓN
ACTORES:	**Administrador, Usuario**
PRECONDICION:	**Es necesario validar usuario, contraseña y rol**
DESCRIPCIÓN:	**Valida el usuario, contraseña y los privilegios de usuario (administrador o usuario), para así poder ingresar a las demás opciones del sistema.**

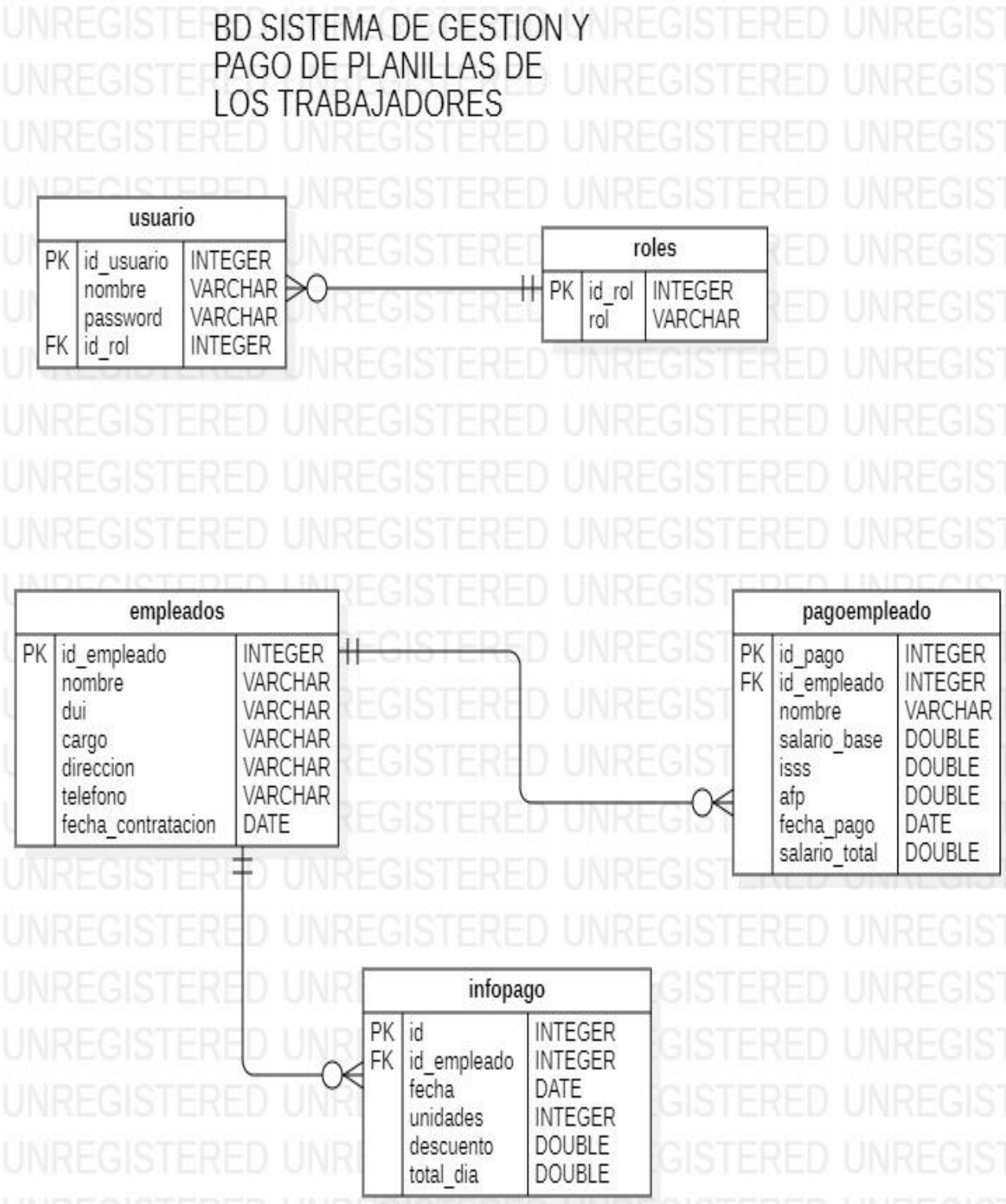
CASO DE USO:	REGISTRAR EMPLEADO
ACTORES:	**Administrador**
PRECONDICION:	**Tener la información personal del empleado**
DESCRIPCIÓN:	**El administrador registra al empleado con su información personal y puede modificar o eliminar sus datos.**

CASO DE USO:	GESTIONAR PAGO AL DIA DEL EMPLEADO
ACTORES:	**Administrador, usuario**
PRECONDICION:	**Tener registrado a los empleados en el sistema**
DESCRIPCIÓN:	**El administrador o el usuario puede registrar lo que ha ganado un empleado al día según la cantidad de unidades elaboradas y este caso dependerá que ya este el empleado registrado en el sistema.**

CASO DE USO:	GESTIONAR PLANILLA
ACTORES:	**Administrador, usuario**
PRECONDICION:	**Tener registrados a los empleados en el sistema**
DESCRIPCIÓN:	**El administrador o el usuario puede registrar una planilla para verificar cuanto es el salario total del empelado en unas fechas estipuladas y este caso dependerá que ya este el empleado registrado en el sistema.**

CASO DE USO:	GENERAR INFORMES
ACTORES:	**Administrador, usuario**
PRECONDICION:	**Tener registrados a los empleados en el sistema y tener planillas registradas en el sistema**
DESCRIPCIÓN:	**El administrador o el usuario pueden generar una ficha de los empleados y un informe de todas las planillas según el mes que inserte el usuario o el administrador.**

5.Diagrama Entidad-Relación



6. Descripción de las tablas.

- Tabla usuario

Nombre del campo.	Tipo de campo.	Descripción.	Tamaño.	Extra
Id_usuario	Autonumérico	Numero unico que se agrega por cada usuario que se registra	11	Primary Key
nombre	Texto	Nombre de usuario ingresado en el sistema.	200	
password	Texto	Contraseña del usuario ingresado en el sistema.	50	
Id_rol	Entero	Campo para relacionar con la tabla roles.	11	Foreing Key

-Tabla roles

Nombre del campo.	Tipo de campo.	Descripción.	Tamaño.	Extra
Id_rol	Entero	Campo que genera número único que ayuda a tener una relación con la tabla usuario	11	Primary Key
rol	Texto	Nombre del rol que muestra el tipo de usuario que se registra	20	

-Tabla empleados

Nombre del campo.	Tipo de campo.	Descripción.	Tamaño.	Extra
Id_empleado	Autonumérico	Campo que genera número único por cada fila que se vaya creando.	11	Primary Key
nombre	Texto	Nombre de cada empleado de la maquila	100	

dui	Texto	Documento único de identidad de cada empleado	12	
cargo	Texto	El tipo de cargo que tiene la persona que desempeña la maquila	100	
direccion	Texto	Direccion de cada empleado que se agrega.	200	
telefono	Texto	Número de contacto	15	
Fecha_contratacion	Fecha/hora	Fecha de contratación	N/A	

-Tabla InfoPago

Nombre del campo.	Tipo de campo.	Descripción.	Tamaño.	Extra
Id	Autonumérico	Campo que genera número único por cada fila que se vaya agregando.	11	Primary Key
Id_empleado	Entero	Id del empleado que se relaciona con la tabla empleados.	11	Foreign Key
fecha	Fecha/hora	Fecha que se registra en la fecha actual de la computadora.	N/A	
unidades	Entero	Unidades que hace el empleado	100	
descuento	Double	Descuento aplicado.	N/A	
Total_dia	Double	Total de dinero que hace cada empleado.	N/A	

-Tabla Pagoempleados

Nombre del campo.	Tipo de campo.	Descripción.	Tamaño.	Extra
Id	Autonumérico	Campo que genera número único por cada fila que se vaya agregando.	11	Primary Key
Id_empleado	Entero	Id del empleado que se relaciona con la tabla empleados.	11	Foreign Key
Nombre	texto	Almacena el nombre de cada empleado	200	
Salario_base	Double	Salario base del empleado	N/A	
isss	Double	Descuento del seguro social	N/A	
afp	Double	Descuento de la administradora de fondos de pensiones.	N/A	
Descuento	Double	Descuento que se le aplica.	N/A	
Fecha_pago	Fecha/hora	Fecha del pago.	N/A	
Salario_total	Double	Salario total del empleado.	N/A	

7. Descripcion de las variables usadas en el sistema

Nombre de la variable	*tipo de variable*	*Descripción de la variable*
con, acceso, conexión, estate	Connection	Se usa más que todo en administrar la conexión
ps	PreparedStatement	Es una sentencia SQL lo cual nos ayuda a ejecutar varias veces una sentencia SQL
Rs, respuesta.	ResultSet	Se usa para llamar a todas las filas que satisfacen las condiciones de una sentencia SQL y proporciona el acceso a los datos de estas filas mediante un conjunto de métodos get que permiten el acceso a las diferentes columnas de las filas.
eu	EntidadUsuario	Se usa para llamar la clase EntidadUsuario atravez del objeto eu.
id_empleado	int	Sirve para guardar un numero único por cada fila de la clase empleados.
nombre	String	Sirve para guardar cada nombre de empleado en la clase empleados
dui	String	Sirve para guardar el documento único de identidad de cada empleado en lo cual lo guarda hacia la clase empleados.
cargo	String	Sirve para guardar el cargo en el cual desempeña cada persona en la maquila en la clase empleados
direccion	String	Sirve para guardar la dirección de cada empleado contratado en la clase empleados
telefono	String	Sirve para guardar el número telefónico de cada empleado contratado en la clase empleados

fecha_contratacion	String	Sirve para guardar la fecha de cada empleado en el cual ha sido contratado, se encuentra en la tabla empleados
id	Int	Guarda un numero único de cada fila, por cada usuario que este registrado en el sistema, se encuentra en la tabla EntidadUsuario
nombre	String	Guarda el nombre de usuario en el cual este registrado para poder ingresar al login, se guarda en la clase EntidadUsuario
password	String	Sirve para guardar la contraseña de cada usuario registrado en el sistema y así poder acceder atravez del login, se guarda en la clase EntidadUsuario
id_rol	String	Sirve para que exista una relación desde la tabla usuarios hacia la tabla rol.
url	String	Almacena la dirección en el cual va a acceder a la base de datos
User	String	Almacena el nombre de usuario en el cual está registrado en el phpMyAdmin se encuentra en la clase ControTrabajadores.
Pass	String	Almacena la contraseña en el cual está registrado en el phpMyAdmin se encuentra en la clase ControTrabajadores.
Id	Int	Guarda un número único por cada fila para la información del pago del trabajador, se encuentra en la clase Infopago.
id_empleado	String	Guarda un número único de cada fila que esta registrado para su información de pago.

Fecha	Date	Almacena la fecha en el cual se le están aplicando los cálculos al pago del empleado.
Unidades	Int	Almacena en la cantidad de unidades crea cada empleado de la maquila, se encuentra en la clase InfoPago.
descuento	Double	Almacena en el descuento en el que se le aplica a cada empleado sino cumple una expectativa en el campo unidades, se encuentra en la clase InfoPago.
total_dia	Double	Almacena la cantidad de producto que hace cada empleado para asi aplicarle su respectivo pago, se encuentra en la clase InfoPago.
Id	Int	Almacena un numero único de cada fila en la planilla.
Idempleado	Int	Guarda el numero único de cada empleado que este en la planilla para poder relacionar con otra tabla, se encuentra en la clase Pago.
salario_base	Double	Almacena el salario en el cual se le va a pagar al empleado esto sin antes aplicarle los descuentos del isss y afp, se encuentra en la clase Pago
Isss	Double	Almacena el descuento en el que se le va a aplicar a cada empleado de la maquila, se encuentra en la clase Pago.
Afp	Double	Almacena el descuento en el que se le va a aplicar a cada empleado de la maquila.
Descuento	Double	Almacena el descuento total en el cual se le va a aplicar el empleado, se encuentra en la clase Pago
Fecha	Date	Almacena la fecha por el cual se puede tener mayor auditoria.

Total	**Double**	**Guarda el total del salario que le toca al empleado.**
total	**Double**	**Guarda el valor de la función calculoISSS y de la función calculoAFP en la clase Pago.**
sqlConsulta	**String**	**Ayuda en guardar la consulta ya sea de insertar, modificar, eliminar etc..**

-Funciones y clases

La clase Conexión sirve para realizar la conexión a través de sus variables base, user, password, url y con su función Conectar () el cual buscara su driver el cual es el conector de mysql.

```
public class Conexion {
     private final String base = "bd_planilla";
    private final String user = "root";
    private final String password = "";
    private final String url = "jdbc:mysql://localhost:3306/" + base;
    Connection con;




    public Connection Conectar(){

         try {
             Class.forName("com.mysql.cj.jdbc.Driver");
             con = DriverManager.getConnection(this.url,this.user,this.password);
            // JOptionPane.showMessageDialog(null, "Conexion exitosa");
         } catch (SQLException e) {
             System.out.println(e);

         } catch (ClassNotFoundException ex) {
           Logger.getLogger(Conexion.class.getName()).log(Level.SEVERE, null, ex);
       }

         return con;

     }


}
```

La clase InfoPago nos ayuda a extraer y enviar datos a la tabla infopago de phpMyAdmin según el cual contiene todos los campos que se necesitan para el seteo y la obtención de datos.

```
public class InfoPago {

    private int id;
    private String id_Empleado;
    private Date fecha;
    private int unidades;
    private Double descuento;
    private Double total_dia;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getId_Empleado() {
        return id_Empleado;
    }

    public void setId_Empleado(String id_Empleado) {
        this.id_Empleado = id_Empleado;
    }

    public Date getFecha() {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    public int getUnidades() {
        return unidades;
    }
```

```
    public void setUnidades(int unidades) {
        this.unidades = unidades;
    }

    public Double getDescuento() {
        return descuento;
    }

    public void setDescuento(Double descuento) {
        this.descuento = descuento;
    }

    public Double getTotal_dia() {
        return total_dia;
    }

    public void setTotal_dia(Double total_dia) {
        this.total_dia = total_dia;
    }


}
```

La clase ConsultaInfoPago nos ayuda en estructurar las consultas necesarias para el sistema y de sus variables ps, rs, sql lo cual se relaciona con la clase Conexión y también de la clase InfoPago para que podamos extraer todas sus variables y ponerlas en práctica en sus siguientes funciones como la función registrar que recibe un objeto de tipo InfoPago. La función registrar ayuda a que podemos insertar datos a la base de datos de phpMyAdmin.

```
public class ConsultaInfoPago extends Conexion {


    public Conexion conn = new Conexion();
     PreparedStatement ps;
     ResultSet rs;


   public boolean registrar(InfoPago p) {


       Connection con = conn.Conectar();

       String sql = "INSERT INTO infopago (id_empleado,fecha,unidades, descuento, total_dia) VALUES(?,?,?,?,?)";

       try {
           ps = con.prepareStatement(sql);
           ps.setInt(1, Integer.parseInt(p.getId_Empleado()));
           ps.setDate(2, p.getFecha());
           ps.setInt(3, p.getUnidades());
           ps.setDouble(4, p.getDescuento());
           ps.setDouble(5, p.getTotal_dia());
           ps.execute();
           return true;
       } catch (Exception e) {

           System.err.println(e);
           return false;
       } finally {

           try {
               con.close();
           } catch (SQLException e) {

               System.err.println(e);
           }

       }

   }
```

La siguiente función es de tipo booleana lo cual retorna verdadero o falso si se cumple o no con la función modificar lo cual nos ayuda en actualizar un dato a la base de datos a través de su variable sql lo cual nos permite realizar la consulta y modificar el dato que queramos realizar.

```
public boolean modificar(InfoPago p) {


    Connection con = conn.Conectar();

     String sql = "UPDATE infopago SET id_empleado=?, fecha=?, unidades=?, descuento=?, total_dia=? WHERE id=?";

     try {
         ps = con.prepareStatement(sql);
         ps.setString(1, p.getId_Empleado());
         ps.setDate(2, p.getFecha());
         ps.setInt(3, p.getUnidades());
         ps.setDouble(4, p.getDescuento());
         ps.setDouble(5, p.getTotal_dia());
         ps.setInt(6, p.getId());
         ps.execute();
         return true;
     } catch (Exception e) {

         System.err.println(e);
         return false;
     } finally {

         try {
             con.close();
         } catch (Exception e) {

             System.err.println(e);
         }

     }

}
```

La funcion eliminar(InfoPago p) recibe un parametro de tipo InfoPago por lo cual esta funcion nos ayuda en eliminar un registro de la base de datos a traves de su id lo busca el registro y lo manda a eliminar.

```
public boolean eliminar(InfoPago p) {


    Connection con = conn.Conectar();

    String sql = "DELETE FROM infopago WHERE id=?";

    try {
        ps = con.prepareStatement(sql);
        ps.setInt(1, p.getId());
        ps.execute();
        return true;
    } catch (SQLException e) {

        System.err.println(e);
        return false;
    } finally {

        try {
            con.close();
        } catch (Exception e) {

            System.err.println(e);
        }

    }

}
```

La función buscar de tipo boolean recibe un parámetro de un objeto de tipo InfoPago, primero hace una consulta que selecciona la tabla en la cual queremos buscar un dato y lo encuentra atravez de su id_empleado luego mandamos a extraer todos los campos y si encuentra el registro retorna verdadero sino falso.

```
public boolean buscar(InfoPago p) {

    Connection con = conn.Conectar();

    String sql = "SELECT * FROM infopago WHERE id_empleado=?";

    try {
        ps = con.prepareStatement(sql);
        ps.setString(1, p.getId_Empleado());
        rs = ps.executeQuery();

        if (rs.next()) {

            p.setId(Integer.parseInt(rs.getString("id")));
            p.setId_Empleado(rs.getString("id_empleado"));
            p.setFecha(Date.valueOf(rs.getString("fecha")));
            p.setUnidades(Integer.parseInt(rs.getString("unidades")));
            p.setDescuento(Double.parseDouble(rs.getString("descuento")));
            p.setTotal_dia(Double.parseDouble(rs.getString("total_dia")));

            return true;
        }

        return false;
    } catch (Exception e) {

        System.err.println(e);
        return false;
    } finally {

        try {
            con.close();
        } catch (Exception e) {

            System.err.println(e);
        }

    }

}
```

La clase Pago ayuda a que tengamos todos los campos de la tabla pagoempleados y luego mandarlos a llamar atravez de su metodo set y get.

```
public class Pago {

    private int id;
    private String idempleado;
    private Double salario_base;
    private Double isss;
    private Double afp;
    private Double descuento;
    private Date fecha;
    private Double total;
    private Date fechainicial;
    private Date fechafinal;
    private String nombre;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Date getFechainicial() {
        return fechainicial;
    }

    public void setFechainicial(Date fechainicial) {
        this.fechainicial = fechainicial;
    }

    public Date getFechafinal() {
        return fechafinal;
    }

    public void setFechafinal(Date fechafinal) {
        this.fechafinal = fechafinal;
    }
```

```
public Date getFechafinal() {
    return fechafinal;
}

public void setFechafinal(Date fechafinal) {
    this.fechafinal = fechafinal;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getIdempleado() {
    return idempleado;
}

public void setIdempleado(String idempleado) {
    this.idempleado = idempleado;
}

public Double getSalario_base() {
    return salario_base;
}

public void setSalario_base(Double salario_base) {
    this.salario_base = salario_base;
}

public Double getIsss() {
    return isss;
}
```

```
public void setIsss(Double isss) {
    this.isss = isss;
}

public Double getAfp() {
    return afp;
}

public void setAfp(Double afp) {
    this.afp = afp;
}

public Double getDescuento() {
    return descuento;
}

public void setDescuento(Double descuento) {
    this.descuento = descuento;
}

public Date getFecha() {
    return fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

public Double getTotal() {
    return total;
}

public void setTotal(Double total) {
    this.total = total;
}
```

La función calculoISSS (Double salario) que recibe un parámetro Double ayuda a sacar el cálculo de seguro social lo cual retorna la variable total

```
public Double calculoISSS(Double salario){

    Double total = 0.0;
    Double isss = 0.03;
    total = salario*isss;
    total = Math.round(total*100.0)/100.0;




return total;
}
```

La función calculoAFP (Double salario) recibe un parámetro Double, ayuda a sacar el cálculo del descuento de la administradora de fondos de pensiones lo cual retorna una variable total

```
public Double calculoAFP(Double salario){

Double total = 0.0;
Double afp = 0.0725;
total = salario*afp;

total = Math.round(total*100.0)/100.0;

return total;
}
```

La clase ConsultaPago depende de la clase Conexión, nos ayuda a realizar las diferentes consultas a la tabla pagoempleados, la función registrar de tipo boolean que recibe un objeto de clase Pago, ayuda a insertar datos atravez de la variable sql.

```
public class ConsultaPago extends Conexion{

    public Conexion conn = new Conexion();
      PreparedStatement ps;
      ResultSet rs;

    public boolean registrar(Pago p) {

        Connection con = conn.Conectar();

        String sql = "INSERT INTO pagoempleados (id_empleado,nombre,salario_base,isss, afp, fecha_pago,salario_total) VALUES(?,?,?,?,?,?,?)";

        try {
            ps = con.prepareStatement(sql);
            ps.setInt(1, Integer.parseInt(p.getIdempleado()));
            ps.setString(2, p.getNombre());
            ps.setDouble(3, p.getSalario_base());
            ps.setDouble(4, p.getIsss());
            ps.setDouble(5, p.getAfp());
            ps.setDate(6, p.getFecha());
            ps.setDouble(7, p.getTotal());

            ps.execute();
            return true;
        } catch (Exception e) {

            System.err.println(e);
            return false;
        } finally {

            try {
                con.close();
            } catch (SQLException e) {

                System.err.println(e);
            }

        }

    }
```

La función modificar () de tipo boolean que recibe un objeto de parte de la clase Pago nos ayuda a modificar un dato a la base de datos a través del id_pago luego modifica cualquier campo solo necesita encontrar el id_pago para realizar la consulta.

```
public boolean modificar(Pago p) {



    Connection con = conn.Conectar();

    String sql = "UPDATE pagoempleados SET id_empleado=?, salario_base=?, isss=?, afp=?, descuento=?, fecha_pago=?, salario_total WHERE id_pago=?";

   try {
        ps = con.prepareStatement(sql);
        ps.setInt(1, Integer.parseInt(p.getIdempleado()));
        ps.setDouble(2, p.getSalario_base());
        ps.setDouble(3, p.getIsss());
        ps.setDouble(4, p.getAfp());
        ps.setDouble(5, p.getDescuento());
        ps.setDate(6, p.getFecha());
        ps.setDouble(7, p.getTotal());
        ps.setInt(7, p.getId());
        ps.execute();
        return true;
    } catch (Exception e) {

        System.err.println(e);
        return false;
    } finally {

        try {
            con.close();
        } catch (SQLException e) {

            System.err.println(e);
        }

    }


}
```

La función eliminar () que recibe el objeto p de tipo Pago ejecuta la consulta de eliminar un dato a través del id_pago en la tabla pagoempleados

```
public boolean eliminar(Pago p) {


    Connection con = conn.Conectar();

     String sql = "DELETE FROM pagoempleados WHERE id_pago=?";

     try {
         ps = con.prepareStatement(sql);
         ps.setInt(1, p.getId());
         ps.execute();
         return true;
     } catch (SQLException e) {

         System.err.println(e);
         return false;
     } finally {

         try {
             con.close();
         } catch (Exception e) {

             System.err.println(e);
         }

     }

}
```

La función salarioBase () de tipo Double recibe un objeto p de tipo Pago esta función nos ayuda a manejar las fechas para asignarle un salarioBase la consulta la hacemos a través de la variable sql que selecciona el campo total_dia de la tabla infopago y que lo encuentra en el id_empleado y la fecha.

```
public Double salarioBase(Pago p){

    Double salariobase = 0.0;
Connection con = conn.Conectar();

     String sql = "SELECT total_dia FROM infopago WHERE id_empleado=? AND fecha BETWEEN ? AND ? ";

    try {
       ps = con.prepareStatement(sql);
       ps.setInt(1, Integer.parseInt(p.getIdempleado()));
       ps.setDate(2, p.getFechainicial());
       ps.setDate(3, p.getFechafinal());

        rs = ps.executeQuery();

        while(rs.next()){


        salariobase = salariobase + Double.parseDouble(rs.getString("total_dia")) ;

        }

        System.out.println(salariobase);
        return salariobase;

    }catch(Exception e){

     System.err.println(e);


    return 0.0;
    }

}
```

La clase empleados utiliza datos que están iguales a los campos de la tabla empleados.

```
public class empleados {

    public int id_empleado;
    public String nombre;
    public String dui;
    public String cargo;
    public String direccion;
    public String telefono;
    public String fecha_contratacion;


}
```

La clase ControTrabajadores hace una conexión para la tabla empleados lo cual trae varias funciones como la función Cargar () de tipo ResultSet lo cual esta funciona ayuda a seleccionar constantemente la tabla empleados.

```
public ControTrabajadores(){
    url="jdbc:mysql://localhost:3306/bd_planilla";
    user = "root";
    pass= "";


    try {
        conexion = DriverManager.getConnection(url,user,pass);
        System.out.println("conexion establecida");
    } catch (SQLException ex) {
        Logger.getLogger(ControTrabajadores.class.getName()).log(Level.SEVERE, null, ex);
    }

}

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        ControTrabajadores p = new ControTrabajadores();
    }

        public ResultSet Cargar(){
    try {
        estate = conexion.createStatement();
        respuesta = estate.executeQuery("SELECT * FROM empleados");
        return respuesta;
    } catch (SQLException ex) {
        Logger.getLogger(ControTrabajadores.class.getName()).log(Level.SEVERE, null, ex);
        return null;
    }
```

La función cerrarConexion () cierra la conexión esto se hace para prevenir inconvenientes de algunas consultas que se dejen abiertas.

```
    public void cerrarConexion(){
try {
    conexion.close();
} catch (SQLException ex) {
    Logger.getLogger(ControTrabajadores.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

La función insertar recibe un objeto p de tipo empleados la variable sqlConsulta ayuda en hacer la consulta de inserción a la tabla empleados y luego pasar a través del objeto p todas las variables que contiene la clase empelados.

```
public boolean insertar(empleados p){

try {
    String sqlConsulta ="INSERT INTO empleados(id_empleado,nombre,dui,cargo,direccion,telefono,fecha_contratacion) VALUES(null,"
            + "'"+p.nombre+"','"+p.dui+"','"+p.cargo+"','"+p.direccion+"','"+p.telefono+"','"+p.fecha_contratacion+"')";
    estate = conexion.createStatement();
    estate.executeUpdate(sqlConsulta);
    return true;
} catch (SQLException ex) {
    Logger.getLogger(ControTrabajadores.class.getName()).log(Level.SEVERE, null, ex);
    return false;
}
}
```

La función editar de tipo boolean recibe un objeto de tipo empleados la variable sqlConsulta hace la consulta UPDATE que significa editar en la tabla empleados y a través del objeto empleados envía todas las variables a la tabla

```
public boolean editar(empleados p){

try {
    String sqlConsulta ="UPDATE empleados SET nombre='"+p.nombre+"',dui='"+p.dui+"',cargo='"+p.cargo+"',"
            + "direccion='"+p.direccion+"',telefono='"+p.telefono+"' WHERE id_empleado="+p.id_empleado;
    estate = conexion.createStatement();
    estate.executeUpdate(sqlConsulta);
    return true;
} catch (SQLException ex) {
    Logger.getLogger(ControTrabajadores.class.getName()).log(Level.SEVERE, null, ex);
    return false;
}
}
```

La función eliminar de tipo boolean recibe un entero id por el cual va a encontrar la fila a eliminar a través de su id_empleado

```
    public boolean Eliminar(int id){

    try {
        String sqlConsulta ="DELETE FROM empleados WHERE id_empleado="+id;
        estate = conexion.createStatement();
        estate.executeUpdate(sqlConsulta);
        return true;
    } catch (SQLException ex) {
        Logger.getLogger(ControTrabajadores.class.getName()).log(Level.SEVERE, null, ex);
        return false;
    }
    }

}
```

La clase EntidadUsuario hace como referencia a la tabla usuario esta clase tiene sus métodos set y get para su posterior uso.

```
public class EntidadUsuario {
    int id;
    String nombre;
    String password;
    String id_rol;

    public EntidadUsuario() {
    }

    public EntidadUsuario(int id, String nombre, String password, String id_rol) {
        this.id = id;
        this.nombre = nombre;
        this.password = password;
        this.id_rol = id_rol;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }
```

```
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getId_rol() {
        return id_rol;
    }

    public void setId_rol(String id_rol) {
        this.id_rol = id_rol;
    }

}
```

La clase CRUD de tipo interface se usa para implementar las funciones que contiene.

```
public interface CRUD {
    public List listar();
    public int add(Object[] o);
    public int actualizar(Object[] o);
    public void eliminar(int id);

}
```

La clase InsertUsuario implementa la interface de clase CRUD para su posterior uso de funciones, la función listar () de tipo List que crea un array para capturar todos los campos que le están seteando el objeto eu, la variable sql hace la consulta que selecciona la tabla usuario.

```
public class InsertUsuario implements CRUD {
    Connection con;
    Conexion cn=new Conexion();
    PreparedStatement ps;
    ResultSet rs;

    @Override
    public List listar() {

        List<EntidadUsuario> lista = new ArrayList<>();
        String sql="SELECT * FROM usuario";
        try {
            con=cn.Conectar();
            ps=con.prepareStatement(sql);
            rs=ps.executeQuery();
            while(rs.next()){
                EntidadUsuario eu = new EntidadUsuario();
                eu.setId(rs.getInt(1));
                eu.setNombre(rs.getString(2));
                eu.setPassword(rs.getString(3));
                eu.setId_rol(rs.getString(4));
                lista.add(eu);
            }
         } catch (Exception e) {
         }
        return lista;
    }
```

La función add de tipo entero recibe un objeto array con lo cual ayuda a guardar los datos obtenidos a través de la consulta sql.

```
public int add(Object[] o) {
    int r=0;
    String sql="INSERT INTO usuario(nombre,password,id_rol) VALUES(?,?,?)";
    try {
        con=cn.Conectar();
        ps=con.prepareStatement(sql);
        ps.setObject(1, o[0]);
        ps.setObject(2, o[1]);
        ps.setObject(3, o[2]);
        r=ps.executeUpdate();
    } catch (Exception e) {
    }
    return r;
}
```

La función actualizar recibe un objeto array el cual le ayuda a obtener varios datos a través de la consulta sql que se utiliza el UPDATE para modificar un campo donde encuentre id_usuario.

```
public int actualizar(Object[] o) {
    int r=0;
    String sql="UPDATE usuario SET nombre=? , password=? , id_rol=? WHERE id_usuario=?";
    try {
        con=cn.Conectar();
        ps=con.prepareStatement(sql);
        ps.setObject(1, o[0]);
        ps.setObject(2, o[1]);
        ps.setObject(3, o[2]);
        ps.setObject(4, o[3]);

        r=ps.executeUpdate();

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

La función eliminar () recibe un entero el cual consta de una variable sql el cual ejecuta DELETE, pero se va ir donde ld_usuario cumpla con la consulta.

```
    public void eliminar(int id) {
        String sql="DELETE FROM usuario WHERE id_usuario=?";
        try {
            con=cn.Conectar();
            ps=con.prepareStatement(sql);
            ps.setInt(1, id);
            ps.executeUpdate();
        } catch (Exception e) {
        }
       ;
    }


}
```

La clase usuarioDao sirve para validar si un usuario es administrador o usuario normal, también si el nombre de usuario y las contraseñas son correctos con los que ya tiene en la BD.

```
public class UsuarioDao {
    PreparedStatement ps;
    ResultSet rs;

    EntidadUsuario eu = new EntidadUsuario();
    Conexion con = new Conexion();
    Connection acceso;

    public EntidadUsuario validarusuario(String password ,String nombre,String id_rol){
        String sql="SELECT * FROM Usuario WHERE password=? and nombre=? and id_rol=?";
        try {
            acceso=con.Conectar();
            ps=acceso.prepareStatement(sql);
            ps.setString(1, password);
            ps.setString(2, nombre);
            ps.setString(3, id_rol);


            rs=ps.executeQuery();
            while (rs.next()) {
                eu.setId(rs.getInt(1));
                eu.setNombre(rs.getString(2));
                eu.setPassword(rs.getString(3));
                eu.setId_rol(rs.getString(4));


            }
        } catch (Exception e) {
        }
        return eu;
    }



}
```

La clase seguridad tiene 2 funciones la primera función que se llama ecnode que recibe una cadena de caracteres en el cual ahí va ir la palabra en que desea encriptar, cabe destacar que esta función retorna la palabra encriptada.

```
public class Seguridad {
    String secretKey = "Seguridad";

    public String ecnode(String cadena) {
        String encriptacion = "";
        try {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            byte[] llavePassword = md5.digest(secretKey.getBytes("utf-8"));
            byte[] BytesKey = Arrays.copyOf(llavePassword, 24);
            SecretKey key = new SecretKeySpec(BytesKey, "DESede");
            Cipher cifrado = Cipher.getInstance("DESede");
            cifrado.init(Cipher.ENCRYPT_MODE, key);
            byte[] plainTextBytes = cadena.getBytes("utf-8");
            byte[] buf = cifrado.doFinal(plainTextBytes);
            byte[] base64Bytes = Base64.encodeBase64(buf);
            encriptacion = new String(base64Bytes);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, "AError al encriptar");
        }
        return encriptacion;
    }
```

La otra función deecnode recibe una cadena encriptada y lo que hace es empezar a desencriptar por lo cual retorna la palabra desencriptada.

```
    public String deecnode(String cadenaEncriptada) {
        String desencriptacion = "";
        try {
            byte[] message = Base64.decodeBase64(cadenaEncriptada.getBytes("utf-8"));
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            byte[] digestOfPassword = md5.digest(secretKey.getBytes("utf-8"));
            byte[] keyBytes = Arrays.copyOf(digestOfPassword, 24);
            SecretKey key = new SecretKeySpec(keyBytes, "DESede");
            Cipher decipher = Cipher.getInstance("DESede");
            decipher.init(Cipher.DECRYPT_MODE, key);
            byte[] plainText = decipher.doFinal(message);
            desencriptacion = new String(plainText, "UTF-8");

        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, "error al desencriptar");
        }
        return desencriptacion;
    }

}
```

8.Diccionario de datos.

Simbologia.

simbolo	Significado
=,:	Simbolo de igualdad y de declaracion de variable
+	Concatenación
[]	Selección una de las opciones encerradas entre corchetes.
{}	Iteraciones del componente encerrado entre llaves.
()	Significa que el componente encerrado es opcional.
//texto,*texto*	Es un comentario

Tabla usuario

Descripcion del sistema de informacion	Sistema de informacion para el manejo y control de planillas de los trabajadores en la Maquila.		
Nombre de la base de datos	Bd_planilla.sql		
Descripcion de la base de datos	Base de datos se encuentra almacenado todo el modelo de gestion de planilla en el sistema.		
Fabricante del DBMS	Servidor Xampp/phpMyAdmin		
Relación:	Esta tabla se relaciona atravez del id_usuario hacia la tabla roles		
Nombre de la tabla	**Nombre del campo.**	**Tipo de dato.**	**Tipo de llave**
usuario	Id_usuario	Int	PK
	Nombre	Varchar	
	Password	Varchar	
	Id_rol	Int	FK

Tabla roles

Descripcion del sistema de informacion	Sistema de informacion para el manejo y control de planillas de los trabajadores en la Maquila.		
Nombre de la base de datos	Bd_planilla.sql		
Descripcion de la base de datos	Base de datos se encuentra almacenado todo el modelo de gestion de planilla en el sistema.		
Fabricante del DBMS	Servidor Xampp/phpMyAdmin		
Relación:	Esta tabla se relaciona atravez del id_rol hacia la tabla usuario		
Nombre de la tabla	**Nombre del campo.**	**Tipo de dato.**	**Tipo de llave**
usuario	Id_rol	Int	PK
	rol	Varchar	

Tabla empleados

Descripcion del sistema de informacion	Sistema de informacion para el manejo y control de planillas de los trabajadores en la Maquila.		
Nombre de la base de datos	Bd_planilla.sql		
Descripcion de la base de datos	Base de datos se encuentra almacenado todo el modelo de gestion de planilla en el sistema.		
Fabricante del DBMS	Servidor Xampp/phpMyAdmin		
Relación:	Esta tabla se relaciona atravez del id_empleado hacia las 2 tablas infopago y pagoempleados.		
Nombre de la tabla	**Nombre del campo.**	**Tipo de dato.**	**Tipo de llave**
empleados	Id_empleado	Int	PK
	nombre	Varchar	
	Dui	Varchar	
	Cargo	varchar	
	Direccion	Varchar	
	Telefono	Varchar	
	Fecha_contratacion	Date	

Tabla pagoempleados

Descripcion del sistema de informacion	Sistema de informacion para el manejo y control de planillas de los trabajadores en la Maquila.		
Nombre de la base de datos	Bd_planilla.sql		
Descripcion de la base de datos	Base de datos se encuentra almacenado todo el modelo de gestion de planilla en el sistema.		
Fabricante del DBMS	Servidor Xampp/phpMyAdmin		
Relación:	Esta tabla se relaciona atravez de la llave foranea del campo id_empleado de la tabla empleados.		
Nombre de la tabla	**Nombre del campo.**	**Tipo de dato.**	**Tipo de llave**
pagoempleados	Id_pago	Int	PK
	Id_empleado	int	FK
	Nombre	Varchar	
	Salario_base	Double	
	Isss	Double	
	Afp	Double	
	Fecha_pago	Date	
	Salario_total	Double	

Tabla infopago

Descripcion del sistema de informacion	Sistema de informacion para el manejo y control de planillas de los trabajadores en la Maquila.		
Nombre de la base de datos	Bd_planilla.sql		
Descripcion de la base de datos	Base de datos se encuentra almacenado todo el modelo de gestion de planilla en el sistema.		
Fabricante del DBMS	Servidor Xampp/phpMyAdmin		
Relación:	Esta tabla se relaciona atravez de la llave foranea del campo id_empleado de la tabla empleados.		
Nombre de la tabla	**Nombre del campo.**	**Tipo de dato.**	**Tipo de llave**
infopago	Id	Int	PK
	Id_empleado	int	FK
	Fecha	Double	
	unidades	Double	

	descuento	Double	
	Total_dia	Double	

Detalles especiales de ejecución

Prerequisitos:

-Xampp version 3.2.4

-Apache Netbeans IDE 12.2

Usos especificos de recursos:

-Una computadora de 4 gb de ram

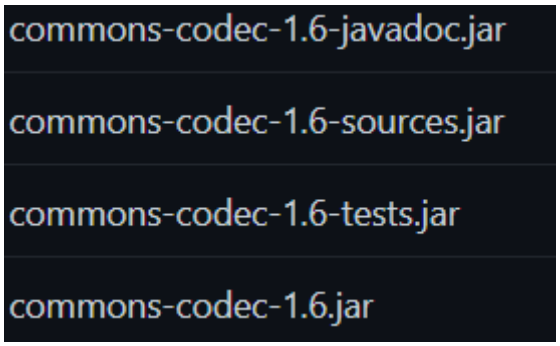
-Procesador intel de 2.4 Ghz.

-Generacion de reportes un aproximado de 3 MB dependiendo de cuanto contenido se introduzca a la base de datos.

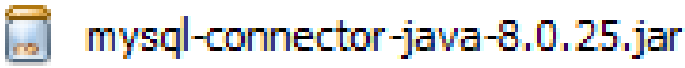
Librerias para generar los reportes:

Itext5-itextpdf-5.5.12.jar

4 Librerias para encryptar y desencriptar las contraseñas y hacer uso de la clase seguridad.java:



Librería para conectar la base de datos:



Descripcion de interfaces con otros sistemas o aplicaciones

No se utilizo otras interfaces externas o aplicaciones.

Bitacoras de cambios dentro de los mismos.

Responsable del cambio	Fecha	Descripcion del cambio
Miguel Bejarano	12-jun-2021	Control de empleados casi terminado
Carlos Posada	18-jun-2021	Manual del programador casi terminado
Gabriel Castillo	17-jun-2021	Diseño y union del login con el manejo de empleados.
Reynaldo	17-jun-2021	Fucion de la rama reynaldo y se hizo transformo a una sola rama con el sistema y control de reportes terminado.