
INSTITUTO TECNOLÓGICO
SUPERIOR ZACATECAS NORTE

INGENIERÍA EN SISTEMAS
COMPUTACIONALES

GESTIÓN DE PROYECTOS DE SOFTWARE

ALUMNO(S):

*JORGE ANTONIO GARCÍA GÓMEZ
KEVIN FABIÁN CRUZ GÓMEZ
CINTHIA ALMARÁZ SIERRA
EDUARDO GARCÍA DELGADO
LUIS ÁNGEL MENDOZA AMAYA*

DOCENTE

MC. DANIEL ARREDONDO SALCEDO

MANUAL TÉCNICO:

GESTIÓN DE DOCUMENTACIÓN DE SERVICIO SOCIAL



CONFECCIÓN	3
II. INTRODUCCIÓN	3
III. OBJETIVOS	3
3.1 OBJETIVO GENERAL	3
3.2 OBJETIVOS ESPECÍFICOS	3
IV. DISEÑO GENERAL	5
V. DISEÑO DETALLADO	6
HU1	6
Diagrama de Clases.	6
Código de la Clase ConexionLuis	7
Código de la Clase ConexionKevin	9
Código de la Clase ConexionCinthia	11
Código de la Clase ConexionEduardo	13
HU1	16
Diagrama de Clases.	16
Código FRM_Login	16
HU2	18
Código de Estudiantes	22
Código de Asesores	28
HU3	33
Diagrama de clases	33
Código IF_AsignacionAsesores	34
HU4	34
Diagrama de clases	34
Código IF_VerDocAdmin	35
HU6	36
Diagrama de clases	36
Código IF_SubirDocumentoEstudiante	37
HU7	38
Diagrama de clases	38
Código IF_VerDescargarDocEstudiante	39
HU8	40
Diagrama de clases	40
Código IF_VerCalificarDocumentacion	41
HU9	43



Diagrama de clases	43
Código IF_EvaluacionEstudiante	43

I. CONFECCIÓN

2.1 Nombre del sistema: Gestión de documentos de servicio social

2.2 Versión del sistema: 1.0

2.3 Tipo de manual: Técnico

2.4 Fecha de elaboración: 21/09/2018

2.5 Área de elaboración: Instituto Tecnológico Superior Zacatecas Norte

II. INTRODUCCIÓN

En el presente manual técnico se trata de recalcar el uso de un sistema para la gestión de documentos de servicio social el cual tiene como objetivo principal la facilitar la centralización de documentos para evitar el uso constante de copias y que se gestione expedientes de servicio social para una mejor organización del mismo servicio social. Agilizando así el control sobre los alumnos en cuanto a su servicio social.


III. OBJETIVOS

3.1 OBJETIVO GENERAL

Facilitar la centralización de documentos para evitar el uso constante de copias y que se gestione expedientes de servicio social basado en el "Lineamiento para la Operación y Acreditación del Servicio Social v1.0" del TecNM.

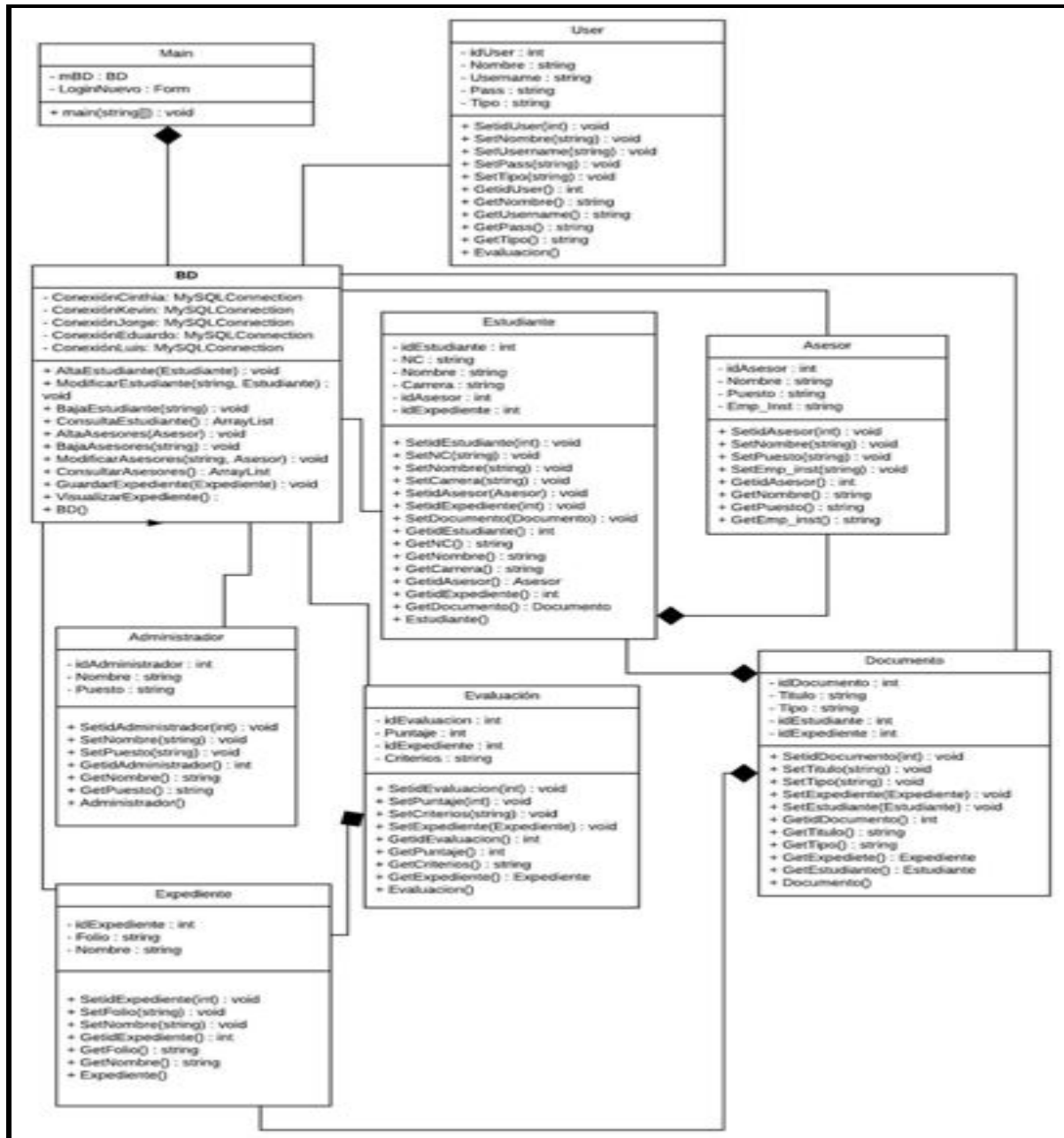
3.2 OBJETIVOS ESPECÍFICOS

1. Tendrá tres tipos de usuario: estudiante, responsables y administrador. Usará un acceso por cuentas usando nombre de usuario y contraseña.
2. Gestión de cuentas: El administrador podrá registrar, modificar, consultar y eliminar las cuentas de estudiantes y asesores.
3. Asignación de asesores: El administrador asignará los estudiantes a un responsable.
4. Consulta de expediente: El administrador podrá visualizar y descargar los documentos proporcionados por un estudiante seleccionado.
5. Reporte de expedientes: El administrador podrá generar los reportes de:
 - Lista de expedientes completados y pendientes.
 - Lista de documentos con su estado (Pendiente, Entregado, Aprobado) por expediente de estudiante.

- 
6. Subir documento. El estudiante podrá subir cada documento correspondiente a su expediente.
 7. Consulta de expediente: El estudiante podrá visualizar y descargar los documentos de su cuenta.
 8. Aprobación de documentos. El responsable o administrador revisarán y marcarán cada documento como aprobado.
 9. Evaluación de Servicio Social: El responsable podrá evaluar a los estudiantes asignados usando el "Anexo I Formato de Evaluación".

IV. DISEÑO GENERAL

Diagrama de clases.



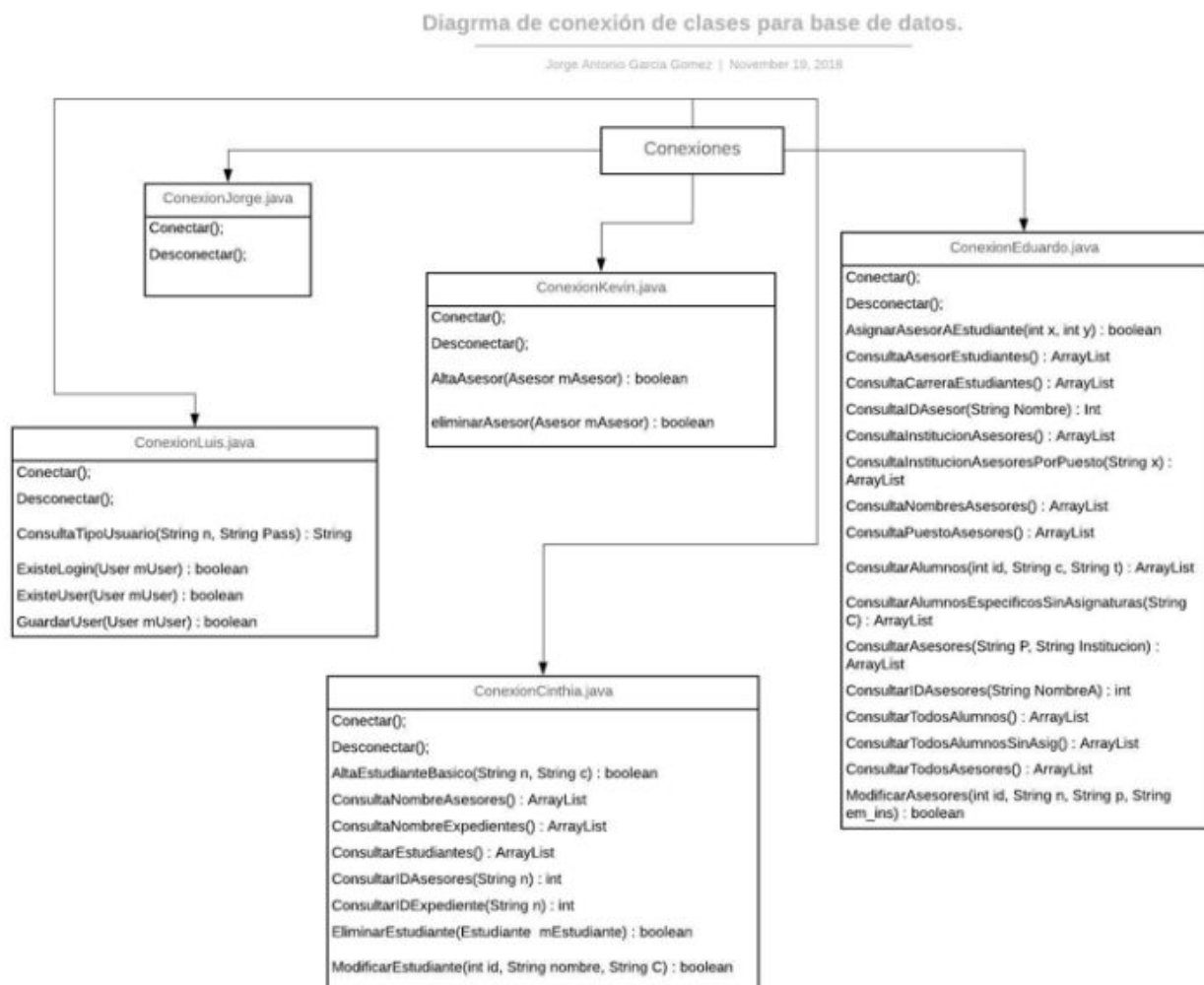
V. DISEÑO DETALLADO

HUI1

Historia de usuario interna para la conexión con la base de datos.

Diagrama de Clases.

Paquete conexiones



Código de la Clase ConexionLuis

```
public String ConsultaTipoUsuario(String Nombre, String Pass) {
    String Tipo = "";
    Statement consulta;
    ResultSet resultado;
    try {
        consulta = conexion.createStatement();
        resultado = consulta.executeQuery("select * from Login where Username = '" +
Nombre + "'and Pass = '" + Pass + "';");
        while (resultado.next()) {
            Tipo = resultado.getString("Tipo");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return Tipo;
}

public Boolean ExisteLogin(User mUser) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
        ResultSet res = consulta.executeQuery("SELECT count(*) as existe FROM Login
WHERE (Pass = '"
        + mUser.getPass() + "' AND Username = '" + mUser.getUsername() + "');");
        if (res.first()) {
            int i = res.getInt("existe");
            if (i > 0) {
                return true;
            }
        }
    }
}
```



```
        } else {
            return false;
        }
    } else {
        return false;
    }
} catch (Exception e) {
    e.printStackTrace();
    return false;
}
}

public Boolean ExisteUser(User mUser) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
        consulta.execute("SELECT Username FROM Login WHERE Username = "
            + mUser.getUsername() + ";;");
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

public boolean GuardarUser(User mUser) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
```

```

        consulta.execute("INSERT INTO Login (Username, Nombre, Pass, Tipo)"
            + "VALUES(" + mUser.getUsername() + ","
            + "" + mUser.getNombre() + ","
            + "" + mUser.getPass() + ","
            + "" + mUser.getTipo() + ");");

        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

```

Código de la Clase ConexionKevin

```

public boolean AltaAsesor(Asesor mAsesor) {
    Statement consulta;

    try {
        consulta = conexion.createStatement();
        consulta.execute("insert into Asesor "
            + "values (null, "
            + "" + mAsesor.getNombre() + ","
            + "" + mAsesor.getPuesto() + ","
            + "" + mAsesor.getEmp_Inst() + ");");

        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

```



```
}
```

```
public boolean eliminarAsesor(Asesor mAsesor) {  
    Statement consulta;  
  
    try {  
        consulta = conexion.createStatement();  
        consulta.execute("delete from Asesor "  
            + " where idAsesor = " + mAsesor.getId_Asesor() + ";"");  
        return true;  
    } catch (Exception e) {  
        e.printStackTrace();  
        return false;  
    }  
}
```

```
public boolean AltaAsesorLogin(String User, String Pass) {  
    Statement consulta;  
  
    try {  
        consulta = conexion.createStatement();  
        consulta.execute("insert into Login "  
            + "values (null, '" + User + "', "  
            + "'" + User + "', "  
            + Pass + "', 'Asesor'" + ");");  
        return true;  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

```
        return false;
    }
}

public int ConsultarIDUltimoAsesor() {
    Statement consulta;
    ResultSet resultado;
    int IDAlumno = 0;

    try {
        consulta = conexion.createStatement();
        resultado = consulta.executeQuery("select idAsesor from Asesor order by idAsesor
DESC limit 1;");
        while (resultado.next()) {
            IDAlumno = resultado.getInt("idAsesor");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return IDAlumno;
}
```

Código de la Clase ConexionCinthia

```
public boolean AltaEstudianteBasico(Estudiante mEstudiante) {
    Statement consulta;

    try {
        consulta = conexion.createStatement();
        consulta.execute("insert into Estudiante "
```

```
        + "values (null,'" + mEstudiante.getNombre() + "','"
        + "'" + mEstudiante.getCarrera() + "','"
        + "null , null,'" + mEstudiante.getNC() + "');"");
    return true;
} catch (Exception e) {
    e.printStackTrace();
    return false;
}
}
```

```
public boolean EliminarEstudiante(Estudiante mEstudiante) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
        consulta.execute("delete from Estudiante "
            + " where NC = " + mEstudiante.getNC() + ";");
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

```
public boolean ModificarEstudiante(String NC, String Nombre, String Carrera) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
```

```

        consulta.execute("update Estudiante set "
            + "Nombre = '" + Nombre + "',"
            + "Carrera = '" + Carrera + "'"
            + " where NC = '" + NC + "';");
        return true;
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error " + e);
        return false;
    }
}

public boolean ModificarExpedienteEstudiante(int idEstudiante, int idExpediente) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
        consulta.execute("update Estudiante set "
            + "Expediente_idExpediente = " + idExpediente
            + " where idEstudiante = " + idEstudiante + ";");
        return true;
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error " + e);
        return false;
    }
}
}

```

Código de la Clase ConexionEduardo

```

public boolean ModificarAsesores(int idAsesor, String NombreAsesorNuevo, String
PuestoAsesorNuevo, String Emp_InstAsesorNuevo) {
    Statement consulta;
    try {

```

```
        consulta = conexion.createStatement();
        consulta.execute("update Asesor set " +
            "Nombre = '" + NombreAsesorNuevo + "', Puesto = '" + PuestoAsesorNuevo + "',
Emp_Inst = '" + Emp_InstAsesorNuevo + "' where idAsesor = '" + idAsesor + "';");
        return true;
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error " + e);
        return false;
    }
}

public boolean ModificarDocumentos(int idDocumento, int idExpediente, int idEstudiante,
String Status) {
    Statement consulta;
    try {
        consulta = conexion.createStatement();
        consulta.execute("update Documento set " +
            "Expediente_idExpediente = '" + idExpediente + "', Estudiante_idEstudiante = '" +
idEstudiante + "', Status= '" + Status + "' where idDocumento = '" + idDocumento + "';");
        return true;
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "Error " + e);
        return false;
    }
}

public byte[] MandarPDF(String NombreDoc) {
    Statement consulta;
    ResultSet resultado;
    byte[] ArrayBites = null;
    try {
```

```
        consulta = conexion.createStatement();

        resultado = consulta.executeQuery("Select Contenido from Documento where Titulo
= " + NombreDoc + ";;");

        while (resultado.next()) {

            ArrayBites = resultado.getBytes("Contenido");

        }

        return ArrayBites;
    } catch (Exception e) {

        JOptionPane.showMessageDialog(null, "Error " + e);

        return ArrayBites;

    }
}

public boolean ModificarEstadoDocumento(String Titulo, String Status) {

    Statement consulta;

    try {

        consulta = conexion.createStatement();

        consulta.execute("update Documento set " +

            "Status= " + Status + " where Titulo = " + Titulo + ";;");

        return true;

    } catch (Exception e) {

        JOptionPane.showMessageDialog(null, "Error " + e);

        return false;

    }

}

public boolean AltaEvaluacion(int Puntaje, int idExpediente) {

    Statement consulta;

    try {

        consulta = conexion.createStatement();
```



```

        consulta.execute("insert into Evaluacion "
            + "values (null, "
            + "'Cuestionario',"
            + "'" + Puntaje + "',"
            + "'" + idExpediente + "');"

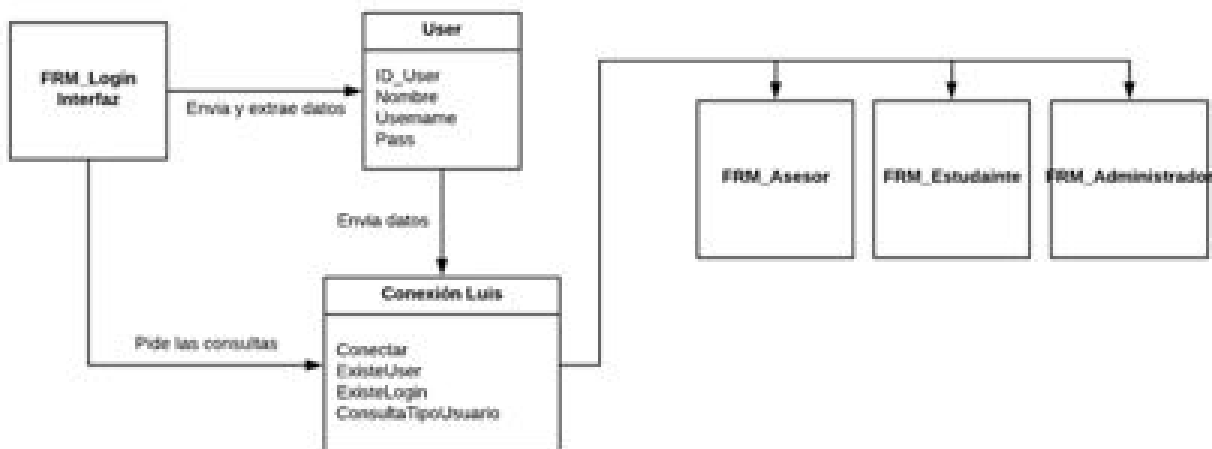
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

```

HU1

Yo como usuario necesito un mecanismo de acceso para realizar mis funciones correspondientes

Diagrama de Clases.



Código FRM_Login

```

private void BtnEntrarActionPerformed(java.awt.event.ActionEvent evt) {
    if (this.TxtUser.getText().isEmpty() || this.TxtPassword.getText().isEmpty()) {

```

```
JOptionPane.showMessageDialog(rootPane, "Asegurese de llenar todos los campos");
} else {
    if (mCL.conectar()) {
        mUser.setUsername(this.TxtUser.getText());
        mUser.setPass(this.TxtPassword.getText());
        Tipo = mCL.ConsultaTipoUsuario(TxtUser.getText(), TxtPassword.getText());
        //JOptionPane.showMessageDialog(rootPane, Tipo);
        if (mCL.ExisteLogin(mUser) && "Administrador".equals(Tipo)) {
            //Administrador
            JOptionPane.showMessageDialog(rootPane, "BIENVENIDO ");
            Usuario = TxtUser.getText();
            this.hide();
            FRM_Administrador mFRM_Administrador = new FRM_Administrador();
            mFRM_Administrador.setVisible(true);
        } else {
            if (mCL.ExisteLogin(mUser) && "Estudiante".equals(Tipo)) {
                //Estudiante
                JOptionPane.showMessageDialog(rootPane, "BIENVENIDO ");
                Usuario = TxtUser.getText();
                this.hide();
                FRM_Estudiante mFRM_Estudiante = new FRM_Estudiante();
                mFRM_Estudiante.setVisible(true);
            } else {
                if (mCL.ExisteLogin(mUser) && "Asesor".equals(Tipo)) {
                    //Asesor
                    JOptionPane.showMessageDialog(rootPane, "BIENVENIDO ");
                    Usuario = TxtUser.getText();
```

```

    Pass = TxtPassword.getText();
    this.hide();

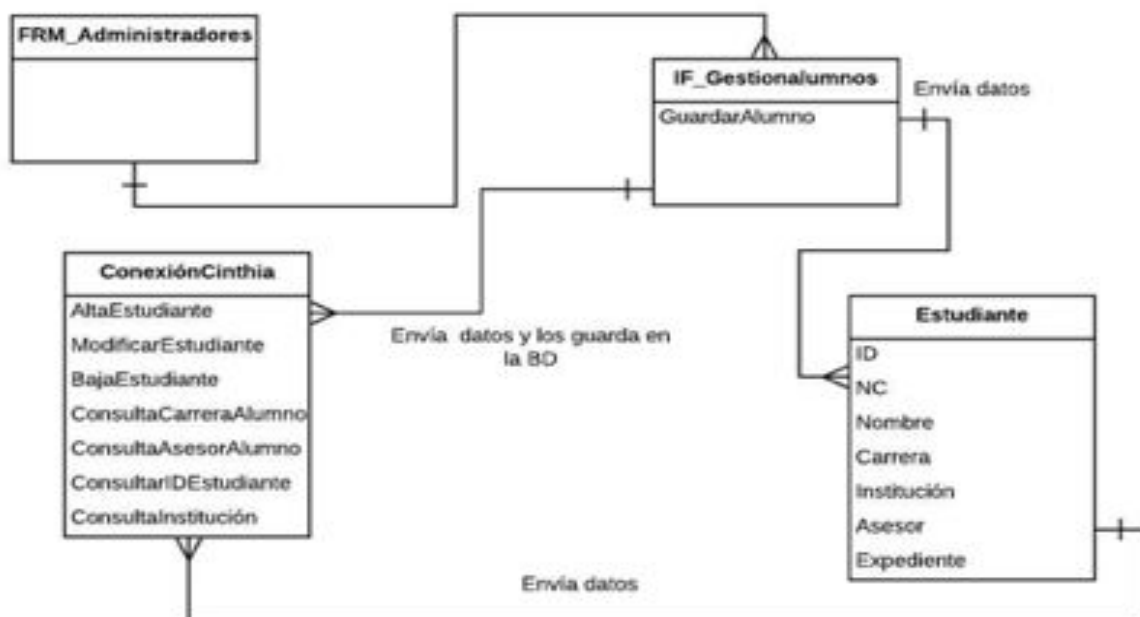
    FRM_Asesor mFRM_Asesor = new FRM_Asesor();
    mFRM_Asesor.setVisible(true);
} else {
    JOptionPane.showMessageDialog(rootPane, "USUARIO INCORRECTO");
}
}
}
}
}
}
}
}
}
}

```

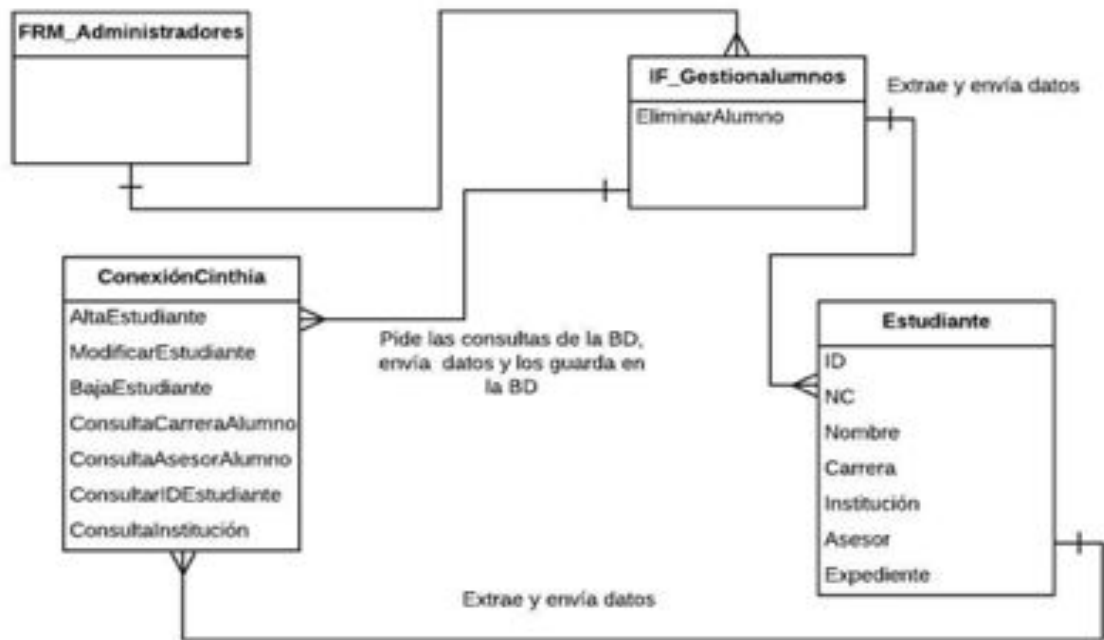
HU2

Yo como administrador del sistema necesito gestionar la información de los estudiantes y asesores con el fin de llevar un control.

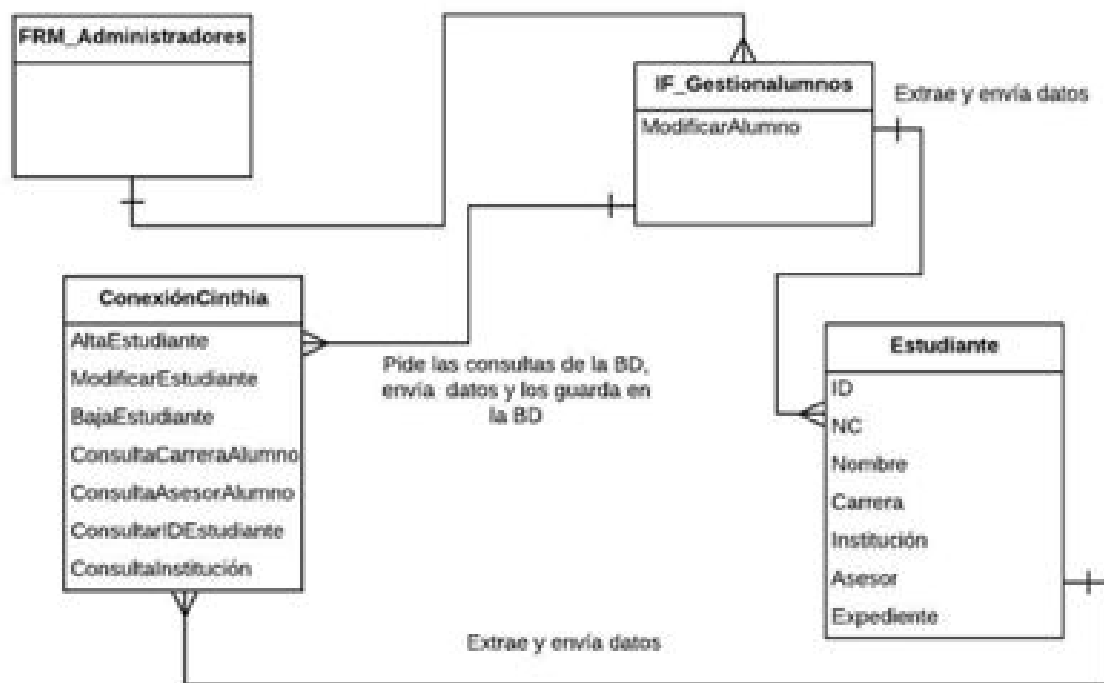
Alta de estudiantes



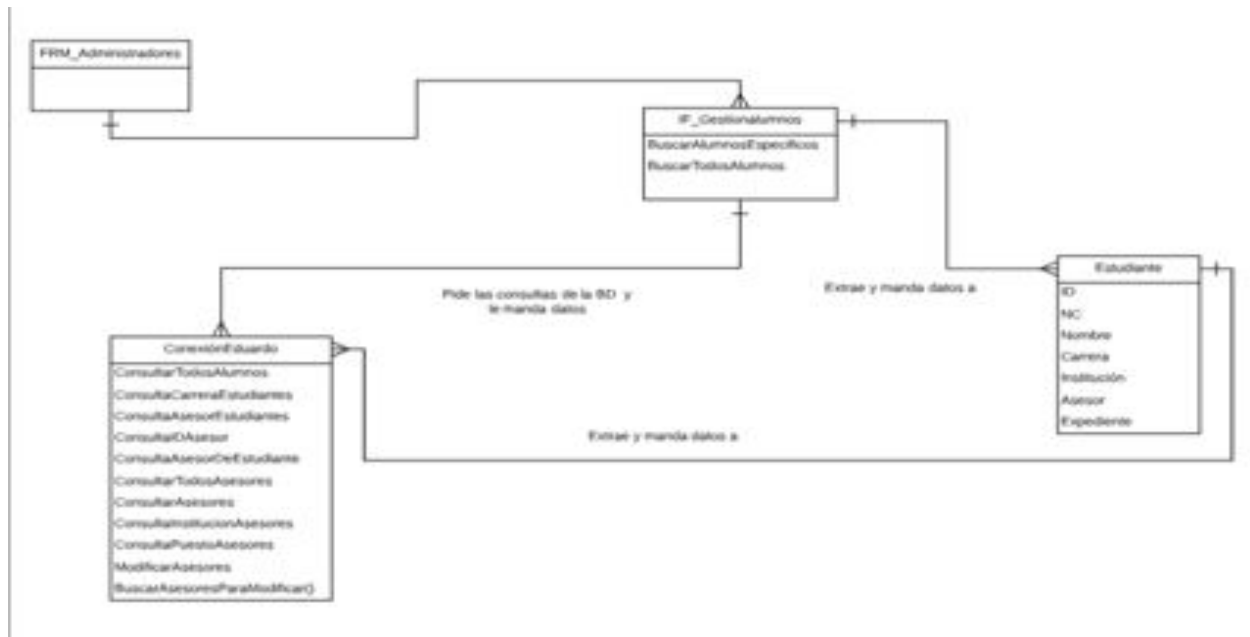
Baja de estudiantes



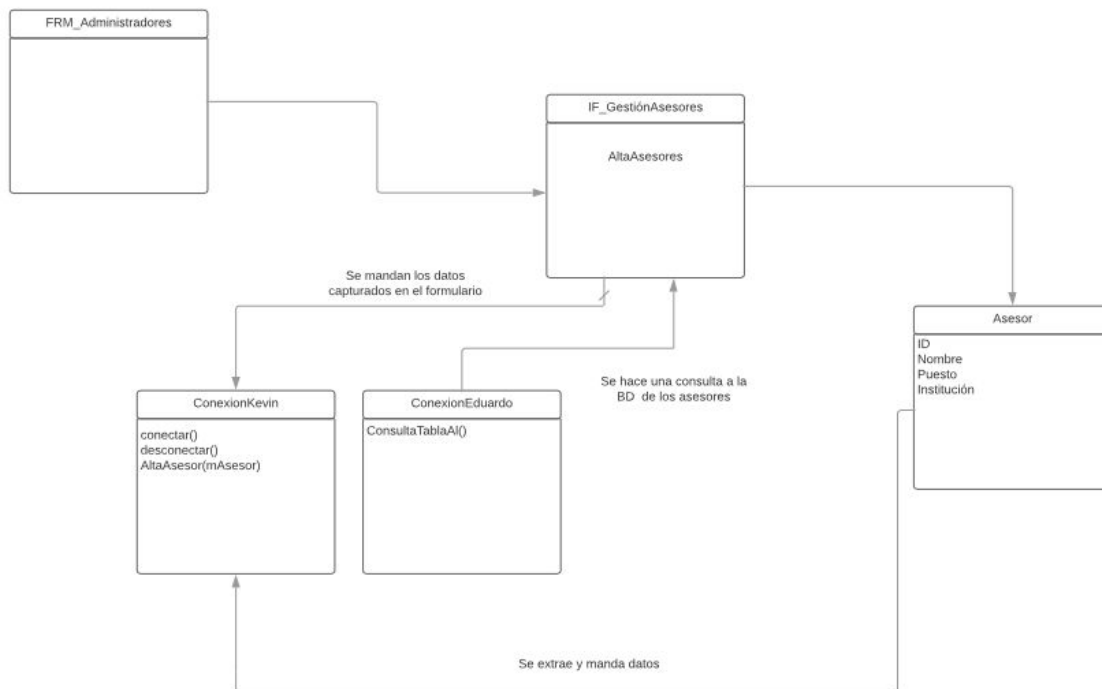
Cambios de estudiantes



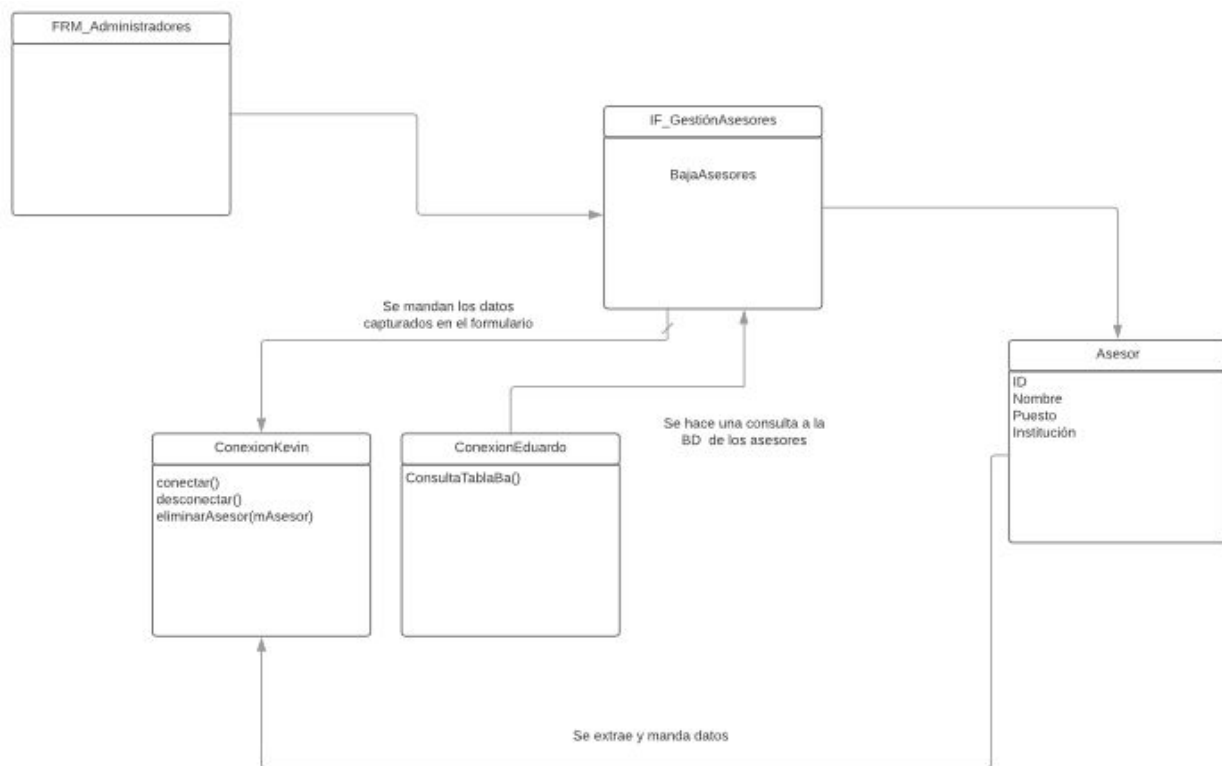
Consulta de estudiantes



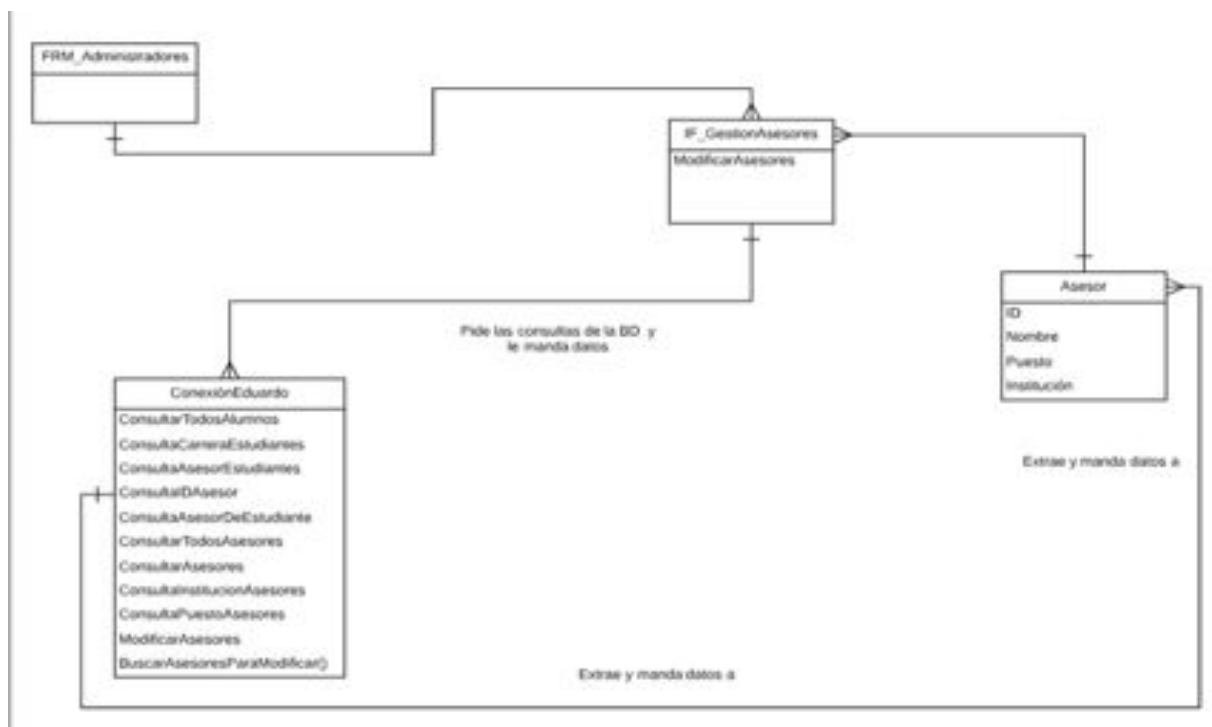
Alta de asesores



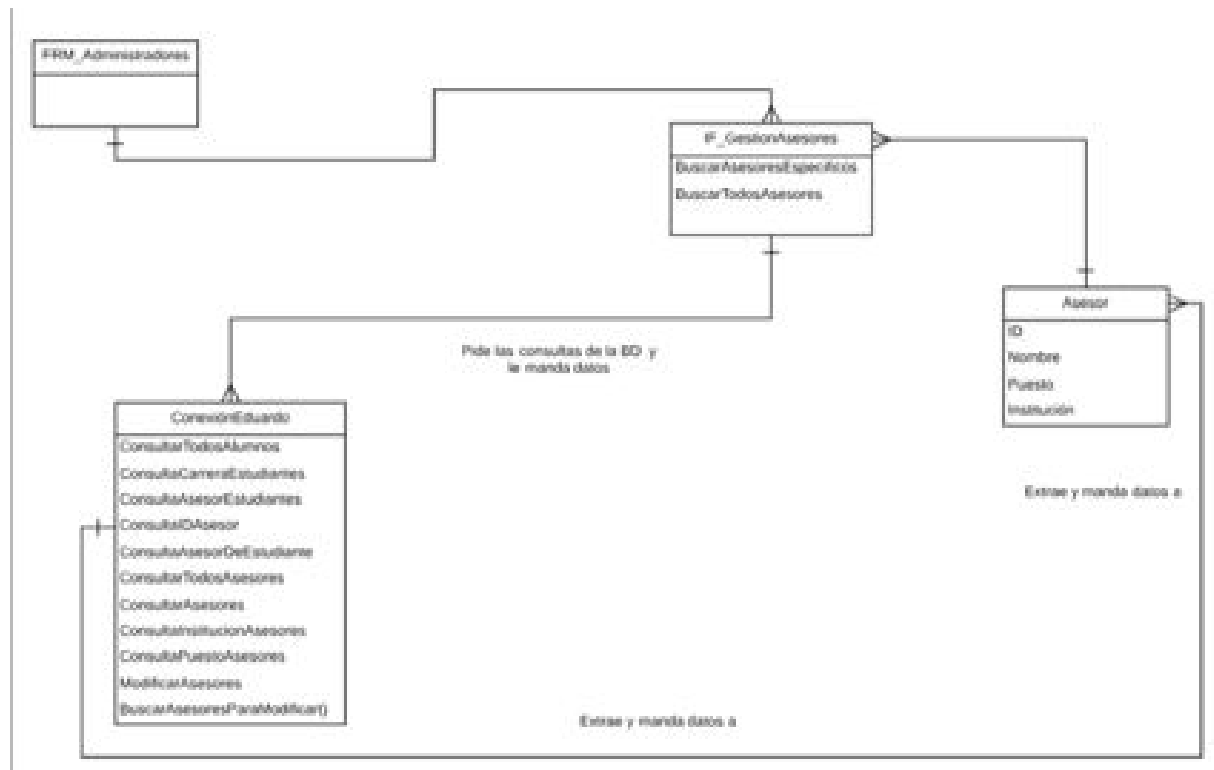
Baja de asesores



Cambios de asesores



Consulta de asesores



Código de Estudiantes

```

public void AltaEstudiante() {
    try {
        Estudiante mEstudiante = new Estudiante();
        String texto = TXTnombreAlta.getText();
        String texto2 = TXTcarreraAlta.getText();
        texto = texto.replaceAll(" ", "");
        texto2 = texto2.replaceAll(" ", "");
        /*
        if (((texto.length() == 0) || (texto.length() >= 150)) && ((texto2.length() == 0) ||
        (texto2.length() >= 150)))) {
            JOptionPane.showMessageDialog(null, "Debe ingresar todos los datos
            correctamente");
        } else {*/
    }
}

```

```

if ((ValidarCajaNombreAltaEst() && ValidarCajaCarreraAltaEst() && validarNC())) {
    if (mCC.conectar()) {
        mEstudiante.setNombre(TXTnombreAlta.getText());
        mEstudiante.setCarrera(TXTcarreraAlta.getText());
        mEstudiante.setNC(TXTnc.getText());
        if (mCC.AltaEstudianteBasico(mEstudiante)) {
            mCC.AltaEstudianteLogin(mEstudiante);
            mCC.AltaEstudianteExpediente(mEstudiante);
            ID_Estudiante = mCC.ConsultarIDAlumnos(mEstudiante.getNC());
            ID_ExpedienteUltimo = mCC.ConsultarIDExpedienteUltimo();
            mCC.ModificarExpedienteEstudiante(ID_Estudiante, ID_ExpedienteUltimo);
            JOptionPane.showMessageDialog(null, "El estudiante fue guardado con
éxito");

            //CBasesorAlta.setSelectedIndex(0);
            //CBexpedienteAlta.setSelectedIndex(0);
            TXTnombreAlta.setText("");
            TXTcarreraAlta.setText("");
            TXTnc.setText("");
        } else {
            JOptionPane.showMessageDialog(null, "Error al guardar al estudiante");
        }
        mCC.desconectar();
        BuscarTodosEstudiantes();
        BuscarTodosEstudiantesBaja();
        BuscarTodosEstudiantesCambio();
        LlenarComboCarrera();
        LlenarComboAsesor();
        TXTnombreAlta.setText("");
    }
}

```



```
        TXTcarreraAlta.setText("");
    }
} else {
    JOptionPane.showMessageDialog(null, "Debe ingresar todos los datos
correctamente");
    //}
}
} catch (HeadlessException | NumberFormatException e) {
    JOptionPane.showMessageDialog(null, "POR FAVOR, LLENE BIEN LOS DATOS");
}
}

public void BajaEstudiante() {
    try {
        if (mCC.conectar()) {
            mEstudiante = new Estudiante();
            mEstudiante.setNC(ID_BajaEst);
            if (mCC.EliminarEstudiante(mEstudiante)) {
                if (!"".equals(ID_BajaEst)) {
                    JOptionPane.showMessageDialog(null, "Estudiante eliminado con éxito");
                } else {
                    JOptionPane.showMessageDialog(null, "Seleccione un estudiante");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Este estudiante no se puede
eliminar,tiene relacion con otros registros");
            }
            mCC.desconectar();
            BuscarTodosEstudiantes();
        }
    }
}
```

```

        BuscarTodosEstudiantesBaja();
        BuscarTodosEstudiantesCambio();
    }
} catch (Exception e) {
    JOptionPane.showMessageDialog(null, "ERROR, Seleccione un estudiante");
}
}

public void ModificarEstudiantes() {
    Estudiante nEstudiante = new Estudiante();
    String texto = TXTnombreCambios.getText();
    String texto2 = TXTcarreraCambios.getText();
    texto = texto.replaceAll(" ", "");
    texto2 = texto2.replaceAll(" ", "");

    if (((texto.length() == 0) || (texto.length() >= 150)) && ((texto2.length() == 0) ||
(texto2.length() >= 150))) {
        JOptionPane.showMessageDialog(null, "Debe ingresar todos los datos
correctamente");
    } else {
        if (mCC.conectar()) {
            //ID = mCC.ConsultarIDAsesores(CBasesorCambios.getSelectedItem().toString());
            //ID2 =
mCC.ConsultarIDExpediente(CBexpedienteCambios.getSelectedItem().toString());
            mEstudiante.setNC(ID_CambioEst);
            nEstudiante.setNombre(this.TXTnombreCambios.getText());
            nEstudiante.setCarrera(this.TXTcarreraCambios.getText());
            nEstudiante.setNC(ID_CambioEst);
            if (ValidarCajaNombreCambioEst() && ValidarCajaCarreraCambioEst()) {
                //nDatosEstudiante.setExpediente_idExpediente(ID2);

```

```
        if (mCC.ModificarEstudiante(ID_CambioEst,TXTnombreCambios.getText(),
TXTcarreraCambios.getText())) {
            JOptionPane.showMessageDialog(null, ID_CambioEst);
            JOptionPane.showMessageDialog(null, "Estudiante modificado
exitosamente");
            LlenarComboCarrera();
            LlenarComboAsesor();
            TXTnombreCambios.setText("");
            TXTcarreraCambios.setText("");
        } else {
            JOptionPane.showMessageDialog(null, "Error al modificar");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Error al llenar campos");
    }
} else {
    JOptionPane.showMessageDialog(null, "Error al conectar con la Base de Datos");
}
mCC.desconectar();
}

BuscarTodosEstudiantes();
BuscarTodosEstudiantesBaja();
BuscarTodosEstudiantesCambio();
}

public void BuscarTodosAlumnos() {
    TablaConsultaEst = (DefaultTableModel) TBAlumnos.getModel();
    int a = TablaConsultaEst.getRowCount() - 1;
    for (int i = a; i >= 0; i--) {
```

```
        TablaConsultaEst.removeRow(TablaConsultaEst.getRowCount() - 1);
    }

    if (mCE.conectar()) {
        ArrayList mArrayListAlumnos = new ArrayList();
        mArrayListAlumnos = mCE.ConsultarTodosAlumnos();
        String[] Datos = null;
        if (mArrayListAlumnos != null) {
            for (int i = 0; i < mArrayListAlumnos.size(); i++) {
                mEstudiante = (Estudiante) mArrayListAlumnos.get(i);
                Datos = new String[3];
                //ID_Asesor = mAsesor.getID_Asesor();
                Datos[0] = mEstudiante.getNombre();
                Datos[1] = mEstudiante.getCarrera();
                Datos[2] = mEstudiante.getNC();
                TablaConsultaEst.addRow(Datos);
            }
        } else {
            //LBL_Mensajero2.setText("No hay puntajes");
        }

        this.TBAlumnos = new javax.swing.JTable();
        this.TBAlumnos.setModel(TablaConsultaEst);
        this.TBAlumnos.getColumnModel().getColumn(0).setPreferredWidth(50);
        this.TBAlumnos.getColumnModel().getColumn(1).setPreferredWidth(50);
        this.TBAlumnos.getColumnModel().getColumn(2).setPreferredWidth(50);

        if (this.TBAlumnos.getRowCount() > 0) {
            this.TBAlumnos.setRowSelectionInterval(0, 0);
        }
    }
}
```

```

    }
} else {
    //LBL_Mensajero2.setText("Error al consultar");
}
mCE.desconectar();
}

```

Código de Asesores

```

public void AgregarAsesor() {
    mAsesor = new Asesor();
    String text = TXTnom.getText();
    //text = text.replaceAll(" ", "");
    if (text.length() != 0 && text.length() <= 35) {
        if (ValidarCajaNombreAlta() && ValidarCajaPuestoAlta() &&
ValidarCajaInstitucionAlta()) {
            if (mCK.conectar()) {
                mAsesor.setNombre(TXTnom.getText());
                String separador = Pattern.quote(" ");
                String[] partes;
                partes = TXTnom.getText().split(separador);
                NombreLoginAsesor = partes[0] + "_" + TXTemp_inst.getText();
                mAsesor.setPuesto(TXTpuesto.getText());
                mAsesor.setEmp_Inst(TXTemp_inst.getText());
                if (mCK.AltaAsesor(mAsesor)) {
                    PassLoginAsesor = String.valueOf(mCK.ConsultarIDUltimoAsesor());
                    mCK.AltaAsesorLogin(NombreLoginAsesor, PassLoginAsesor);
                    JOptionPane.showMessageDialog(null, "Asesor agregado exitosamente \n"
                        + "Usuario: " + NombreLoginAsesor + "\n"
                        + "Contraseña: " + PassLoginAsesor + "");
                }
            }
        }
    }
}

```

```
        } else {
            JOptionPane.showMessageDialog(null, "Error al guardar Asesor");
        }
    }
    mCE.desconectar();
} else {
    JOptionPane.showMessageDialog(null, "Escribe un nombre");
}
} else {
    JOptionPane.showMessageDialog(null, "Nombre no válido");
}
ConsultaTablaAl();
LlenarComboPuestoConsulta();
LlenarComboInstitucionConsulta();
LlenarComboPuesto();
LlenarComboInstitucion();
ConsultaTablaBa();
TXTnom.setText("");
TXTpuesto.setText("");
TXTemp_inst.setText("");
//BuscarAsesoresParaModificar();
}

public void EliminarAsesor() {
    String text = TXTnom3.getText();
    text = text.replaceAll(" ", "");
    if (text.length() != 0 && text.length() <= 35) {
```

```
if (mCK.conectar()) {
    if (ValidarCajaNombreBaja()) {
        mAsesor.setId_Asesor(ID_AsesorBaja);
        if (mCK.eliminarAsesor(mAsesor)) {
            JOptionPane.showMessageDialog(null, "Asesor dado de baja exitosamente");
        } else {
            JOptionPane.showMessageDialog(null, "Este Asesor tiene relación con otros
registros");
        }
    } else {
        JOptionPane.showMessageDialog(null, "Selecciona el Asesor que deseas
eliminar");
    }
} else {
    JOptionPane.showMessageDialog(null, "Error al conectar");
}
} else {
    JOptionPane.showMessageDialog(null, "Nombre no válido");
}
mCE.desconectar();
ConsultaTablaAl();
LlenarComboPuestoConsulta();
LlenarComboInstitucionConsulta();
LlenarComboPuesto();
LlenarComboInstitucion();
ConsultaTablaBa();
TXTnom3.setText("");
}
```

```
public void ModificarAsesores() {  
    if (ValidarCajaNombre() && ValidarCajaPuesto() && ValidarCajaInstitucion()) {  
        if (mCE.conectar()) {  
            if (mCE.ModificarAsesores(ID_Asesor, TXTNombreNuevo.getText(),  
TXTPuestoNuevo.getText(), TXTInstitucionNueva.getText())) {  
                //LBL_Mensajero2.setText("Se realizó un cambio");  
                JOptionPane.showMessageDialog(null, "Cambio realizado");  
                BuscarAsesoresParaModificar();  
                mCE.desconectar();  
            } else {  
                //no se hizo modificacion  
                JOptionPane.showMessageDialog(null, "No se realizó el cambio");  
            }  
        } else {  
            //no conectado  
            JOptionPane.showMessageDialog(null, "No conectado a la BD");  
        }  
    } else {  
        //llenar bien campos  
        JOptionPane.showMessageDialog(null, "Llenar bien los campos");  
    }  
    ConsultaTablaAl();  
    LlenarComboPuestoConsulta();  
    LlenarComboInstitucionConsulta();  
    LlenarComboPuesto();  
    LlenarComboInstitucion();  
    ConsultaTablaBa();  
    TXTNombreNuevo.setText("");  
}
```



```
        TXTPuestoNuevo.setText("");
        TXTInstitucionNueva.setText("");
    }
    public void BuscarTodosAsesores() {
        TablaConsulta = (DefaultTableModel) TBAsesores1.getModel();
        int a = TablaConsulta.getRowCount() - 1;
        for (int i = a; i >= 0; i--) {
            TablaConsulta.removeRow(TablaConsulta.getRowCount() - 1);
        }
        if (mCE.conectar()) {
            ArrayList mArrayListAsesores = new ArrayList();
            mArrayListAsesores = mCE.ConsultarTodosAsesores();
            String[] Datos = null;
            if (mArrayListAsesores != null) {
                for (int i = 0; i < mArrayListAsesores.size(); i++) {
                    mAsesor = (Asesor) mArrayListAsesores.get(i);
                    Datos = new String[3];
                    ID_Asesor = mAsesor.getId_Asesor();
                    Datos[0] = mAsesor.getNombre();
                    Datos[1] = mAsesor.getPuesto();
                    Datos[2] = mAsesor.getEmp_Inst();
                    TablaConsulta.addRow(Datos);
                }
            } else {
                //LBL_Mensajero2.setText("No hay puntajes");
            }
            this.TBAsesores1 = new javax.swing.JTable();
        }
    }
}
```

```

this.TBAseores1.setModel(TablaConsulta);

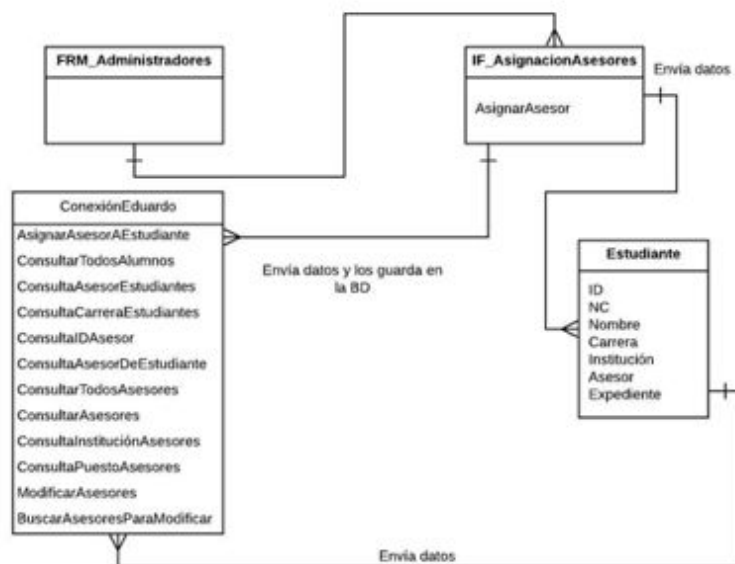
this.TBAseores1.getColumnModel().getColumn(0).setPreferredWidth(50);
this.TBAseores1.getColumnModel().getColumn(1).setPreferredWidth(50);
this.TBAseores1.getColumnModel().getColumn(2).setPreferredWidth(100);
if (this.TBAseores1.getRowCount() > 0) {
    this.TBAseores1.setRowSelectionInterval(0, 0);
}
} else {
    //LBL_Mensajero2.setText("Error al consultar");
}
mCE.desconectar();
}

```

HU3

Yo como administrador quiero asignar un asesor a los estudiantes esto con el fin de seguir el lineamiento basado en el TecNM.

Diagrama de clases



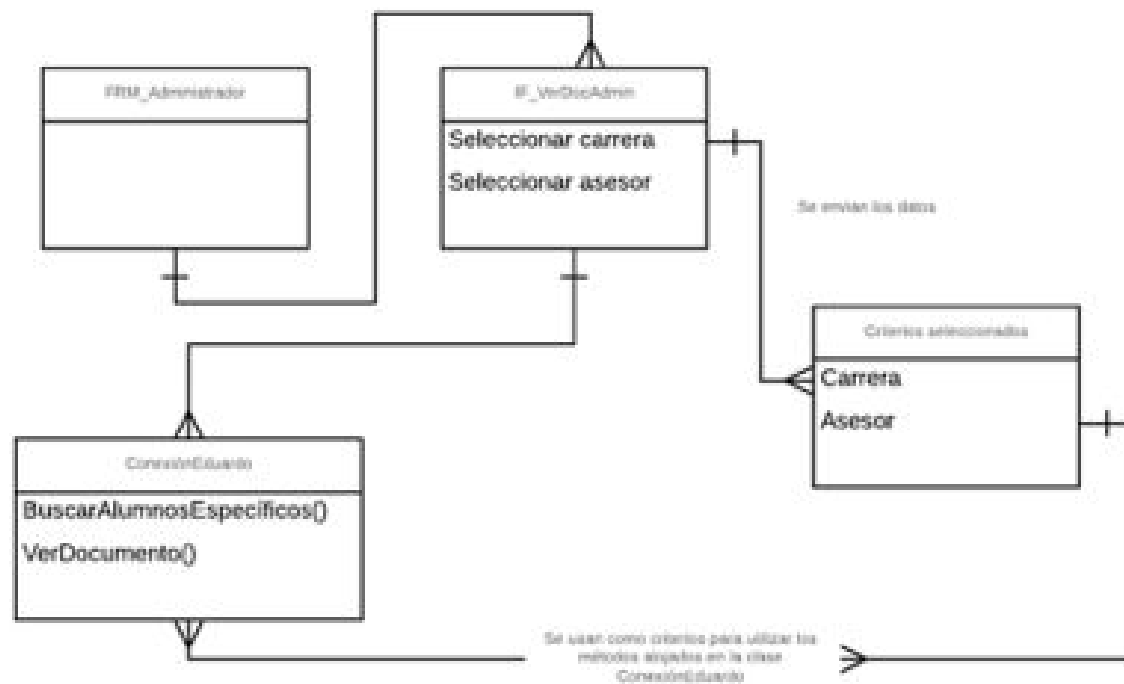
Código IF_AsignacionAsesores

```
public void AsignarAsesor() {  
    if (mCE.conectar()) {  
        ID_AsesorAsignado =  
mCE.ConsultaIDAsesor(CBAsesoresAsignar.getSelectedItem().toString());  
        if (mCE.AsignarAsesorAEstudiante(ID_AlumnoParaAsesor, ID_AsesorAsignado)) {  
            //LBL_Mensajero2.setText("Se realizó un cambio");  
            JOptionPane.showMessageDialog(null, "Cambio realizado");  
            BuscarTodosAlumnos();  
            mCE.desconectar();  
        } else {  
            //no se hizo modificacion  
            JOptionPane.showMessageDialog(null, "No se realizó el cambio");  
        }  
    } else {  
        //no conectado  
        JOptionPane.showMessageDialog(null, "No conectado a la BD");  
    }  
}
```

HU4

Yo como administrador y/o asesor quiero visualizar y descargar los documentos proporcionados por el estudiante con la finalidad de poder checarlos en la plataforma.

Diagrama de clases



Código IF_VerDocAdmin

```

public void VerDocumento() {
    String ArchivoPDF = "PruebaDoc.pdf";
    boolean correcto = false;
    boolean SO = true;
    try {
        out = new FileOutputStream(ArchivoPDF);
        if (mCE.conectar()) {
            String separador = Pattern.quote("");
            String[] partes;
            partes = CBTitulosDoc.getSelectedItem().toString().split(separador);
            out.write(mCE.MandarPDF(partes[0]));
            mCE.desconectar();
            out.close();
        }
    }
}

```

```

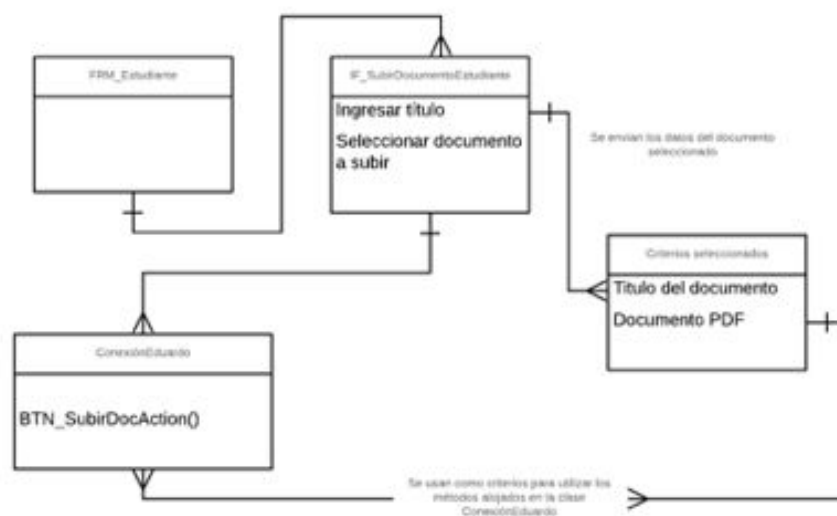
correcto = true;
try {
    Runtime.getRuntime().exec("open " + ArchivoPDF);
} catch (IOException e) {
    e.printStackTrace();
    Runtime.getRuntime().exec("start \"\" /max " + ArchivoPDF + "\""); //pal
Windows
}
} else {
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

HU6

Yo como estudiante deseo subir un documento conforme a mi expediente para mantener a mi asesor o al administrador informados de mi avance.

Diagrama de clases



Código IF_SubirDocumentoEstudiante

```
private void BTN_SubirDocActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String Titulo = TXTTitulo.getText();  
    String Tipo = "PDF";  
    String Ruta = TXTRuta.getText();  
    sql s = new sql();  
    int codigo = s.auto_increment("SELECT MAX(idDocumento) FROM Documento;");  
    File ruta = new File(TXTRuta.getText());  
    if (Titulo.trim().length() != 0 && TXTRuta.getText().trim().length() != 0 ) {  
        guardar_pdf(idDocumento, Titulo, Tipo, ruta);  
        if (mCE.conectar()) {  
            idEstudiante = mCE.ConsultarIDAlumnos(mFL.Usuario);  
            idExpediente = mCE.ConsultarIDExpediente("Expediente_"+mFL.Usuario);  
            idUltimoDoc = mCE.ConsultarIDUltimoDoc();  
            Status = "No revisado";  
            mCE.ModificarDocumentos(idUltimoDoc, idExpediente, idEstudiante, Status);  
            mCE.desconectar();  
        } else {  
            JOptionPane.showMessageDialog(null, "Conectar con la Base de Datos");  
        }  
        //tpdf.visualizar_PdfVO(tabla);  
        TXTRuta.setText("");  
        //activa_boton(false, false, false);  
        TXTTitulo.setText("");  
        //JOptionPane.showMessageDialog(null, "Archivo subido con exito");  
    } else {
```

```

        JOptionPane.showMessageDialog(null, "Rellenar todo los campos");
    }
}

private void TXTTituloActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

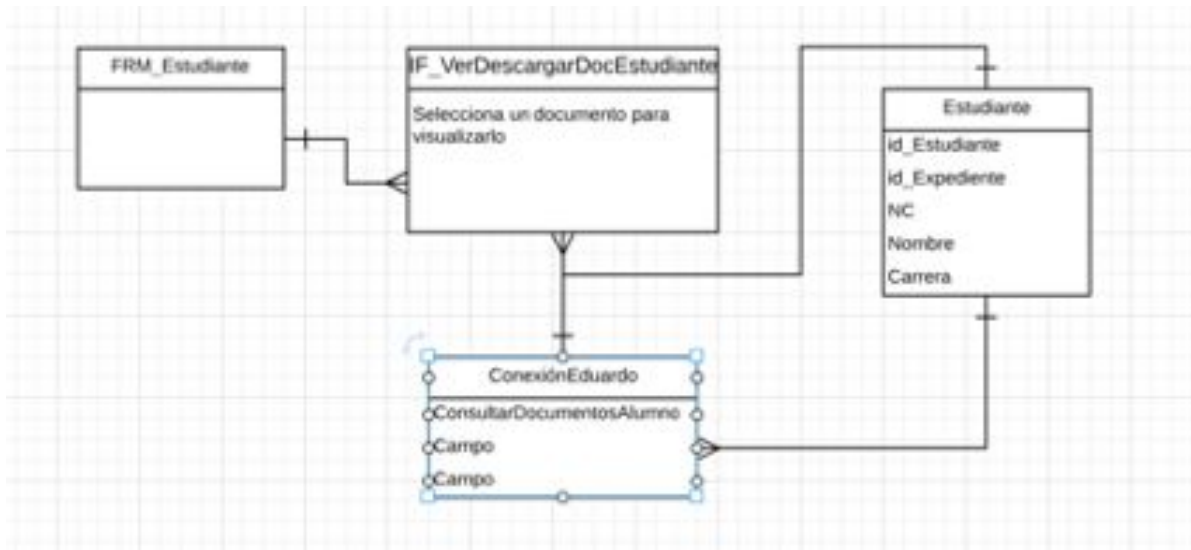
public void guardar_pdf(int idDoc, String titulo, String tipo, File ruta) {
    PdfDAO pa = new PdfDAO();
    PdfVO po = new PdfVO();
    po.setId(idDoc);
    po.setTitulopdf(titulo);
    po.setTipopdf(tipo);
    try {
        byte[] pdf = new byte[(int) ruta.length()];
        InputStream input = new FileInputStream(ruta);
        input.read(pdf);
        po.setArchivopdf(pdf);
    } catch (IOException ex) {
        po.setArchivopdf(null);
        JOptionPane.showMessageDialog(null, "Error al agregar archivo pdf
"+ex.getMessage());
    }
    pa.Agregar_PdfVO(po);
}

```

HU7

Yo como estudiante deseo visualizar y/o descargar los documentos a mi cuenta con la finalidad de poder corregirlos si es necesario o solicitado.

Diagrama de clases



Código IF_VerDescargarDocEstudiante

```

public void VerDocumento() {
    String ArchivoPDF = "PruebaDoc.pdf";
    boolean correcto = false;
    boolean SO = true;
    try {
        out = new FileOutputStream(ArchivoPDF);
        if (mCE.conectar()) {
            String separador = Pattern.quote("(");
            String[] partes;
            partes = CBTitulosDoc.getSelectedItem().toString().split(separador);
            out.write(mCE.MandarPDF(partes[0]));
            mCE.desconectar();
            out.close();
            correcto = true;
        }
        try {
            Runtime.getRuntime().exec("open " + ArchivoPDF);
        }
    }
}

```



```

    } catch (IOException e) {
        e.printStackTrace();

        Runtime.getRuntime().exec("start \\\" /max \" + ArchivoPDF + "\\"); //pal
Windows

    }

    } else {

    }

    } catch (Exception e) {
        e.printStackTrace();
    }

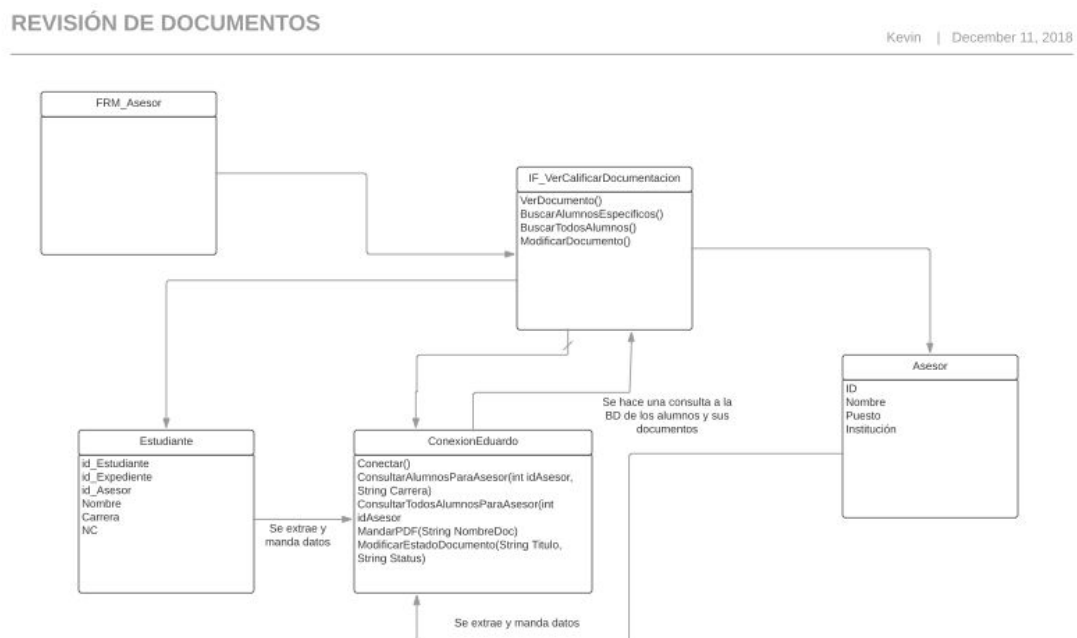
}

```

HU8

Yo como asesor y/o administrador necesito revisar y marcar los documentos como “aprobados” para que el alumno se informe de su avance.

Diagrama de clases



Código IF_VerCalificarDocumentacion

```

public void VerDocumento() {
    String ArchivoPDF = "PruebaDoc.pdf";
    boolean correcto = false;
    boolean SO = true;
    try {
        out = new FileOutputStream(ArchivoPDF);
        if (mCE.conectar()) {
            String separador = Pattern.quote("(");
            String[] partes;
            partes = CBTitulosDoc.getSelectedItem().toString().split(separador);
            out.write(mCE.MandarPDF(partes[0]));
            mCE.desconectar();
            out.close();
            correcto = true;
            try {
                Runtime.getRuntime().exec("open " + ArchivoPDF);
            } catch (IOException e) {
                e.printStackTrace();
                Runtime.getRuntime().exec("start \"%\" /max " + ArchivoPDF + "\""); //pal
Windowsds
            }
        } else {

        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

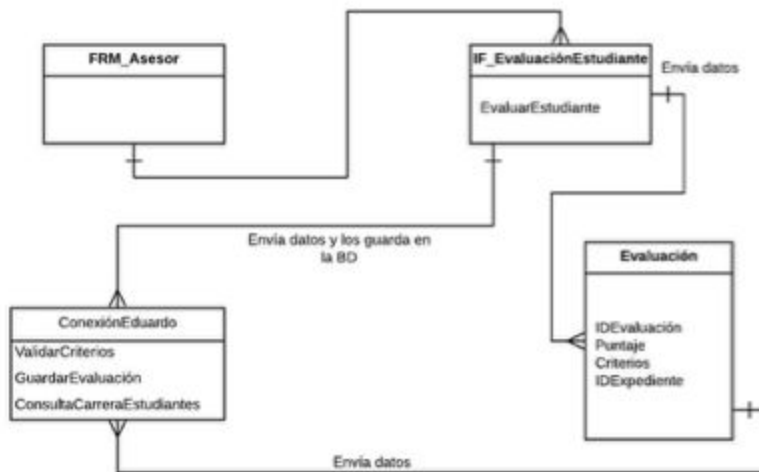
```

```
}  
public void ModificarDocumento() {  
    if (CBTitulosDoc.getItemCount() != 0) {  
        if (mCE.conectar()) {  
            if (mCE.ModificarEstadoDocumento(LBL_NombreDoc.getText(),  
CB_Status.getSelectedItem().toString())) {  
                //LBL_Mensajero2.setText("Se realizó un cambio");  
                JOptionPane.showMessageDialog(null, "Cambio realizado");  
                LBL_NombreDoc.setText("-----");  
                CB_Status.setSelectedIndex(0);  
                ActualizarCBTitulos();  
                LlenarComboTitulos();  
                mCE.desconectar();  
            } else {  
                //no se hizo modificacion  
                JOptionPane.showMessageDialog(null, "No se realizó el cambio");  
            }  
        } else {  
            //no conectado  
            JOptionPane.showMessageDialog(null, "No conectado a la BD");  
        }  
    } else {  
        //llenar bien campos  
        JOptionPane.showMessageDialog(null, "Llenar bien los campos");  
    }  
}
```

HU9

Yo como asesor y/o administrador quiero evaluar a los estudiantes usando el “Anexo I Formato de Evaluación” con el fin de seguir el protocolo de evaluación requerido.

Diagrama de clases



Código IF_EvaluacionEstudiante

```

public void VerDocumento() {
    String ArchivoPDF = "PruebaDoc.pdf";
    boolean correcto = false;
    boolean SO = true;
    try {
        out = new FileOutputStream(ArchivoPDF);
        if (mCE.conectar()) {
            String separador = Pattern.quote("(");
            String[] partes;
            partes = CBTitulosDoc.getSelectedItem().toString().split(separador);
            out.write(mCE.MandarPDF(partes[0]));
        }
    }
}
  
```

```
mCE.desconectar();
out.close();
correcto = true;
try {
    Runtime.getRuntime().exec("open " + ArchivoPDF);
} catch (IOException e) {
    e.printStackTrace();
    Runtime.getRuntime().exec("start \"" /max " + ArchivoPDF + "\""); //pal
Windowds
    }
} else {
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```