## UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO

## **Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones**



**Proyecto Final**

**INTEGRANTES:**

**MATERIA:** Ing. de Calidad

**Santa Cruz – Bolivia**

**CONTENIDO**

[**1.** **TITULO DEL TEMA** 3](#_Toc66374586)

[**2.** **DESRIPCIÓN DEL PROYECTO** 3](#_Toc66374587)

[**3.** **DEFINICIÓN DE REQUERIMIENTOS CON LA NORMA IEEE830** 3](#_Toc66374588)

[**3.1.** **Introducción** 3](#_Toc66374589)

[**3.1.1.** **Propósito** 3](#_Toc66374590)

[**3.1.2.** **Ámbito del Sistema** 3](#_Toc66374591)

[**3.1.3.** **Definiciones, Acrónimos y Abreviaturas** 4](#_Toc66374592)

[**3.1.4.** **Referencias** 4](#_Toc66374593)

[**3.1.5.** **Visión General del Documento** 4](#_Toc66374594)

[**3.2.** **Descripción General** 4](#_Toc66374595)

[**3.2.1.** **Perspectiva del Producto** 4](#_Toc66374596)

[**3.2.2.** **Funciones del producto** 4](#_Toc66374597)

[**3.2.3.** **Características de los usuarios** 5](#_Toc66374598)

[**3.2.4.** **Restricciones** 5](#_Toc66374599)

[**3.2.5.** **Suposiciones y Dependencias** 5](#_Toc66374600)

[**3.2.6.** **Requisitos Futuros** 5](#_Toc66374601)

[**3.3.** **Requisitos Específicos** 5](#_Toc66374602)

[**3.3.1.** **Interfaces Externas** 7](#_Toc66374603)

[**3.3.2.** **Funciones** 7](#_Toc66374604)

[**3.3.3.** **Requisitos de Rendimiento** 7](#_Toc66374605)

[**3.3.4.** **Restricciones de Diseño** 7](#_Toc66374606)

[**3.3.5.** **Atributos del Sistema** 7](#_Toc66374607)

[**3.3.6.** **Otros Requisitos** 7](#_Toc66374608)

[**4.** **ANÁLISIS** 7](#_Toc66374609)

[**5.** **DISEÑO** 7](#_Toc66374610)

[**6.** **DESARROLLO** 7](#_Toc66374611)

[**7.** **PRUEBAS** 7](#_Toc66374612)

1. **TITULO DEL TEMA**

Programa para el registro de notas.

1. **DESRIPCIÓN DEL PROYECTO**

Este proyecto consta de un programa implementado para el registro de notas de estudiantes, y será utilizado por profesores y docentes que requieren del sistema para tener un mejor control de las notas de sus estudiantes.

1. **DEFINICIÓN DE REQUERIMIENTOS CON LA NORMA IEEE830**
   1. **Introducción**

Este documento es una Especificación de Requisitos de Software para el sistema de información para registro de notas de estudiantes. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar IEEE830.

* + 1. **Propósito**

El presente documento tiene como propósito definir los requerimientos funcionales y no funcionales para el desarrollo de un sistema de registro de notas que permita gestionar de una manera más eficaz este proceso. Este será utilizado por profesores y docentes.

* + 1. **Ámbito del Sistema**

Se implementará los requerimientos más básicos y esenciales como ser:

* El usuario podrá registrar a los estudiantes a sus diferentes materias.
* Se podrá asignar nota a cada estudiante de acuerdo a las diferentes materias que se tiene registrado.
* Se podrá observar un listado de los estudiantes con las diferentes materias que tiene registrado cada uno de los estudiantes, y con su respectiva nota asignada.
  + 1. **Definiciones, Acrónimos y Abreviaturas**

Usuario: Persona que usa el sistema para gestionar procesos.

Programa: Es una secuencia de instrucciones, escritas para realizar una tarea específica en una computadora.

ERS: Especificación de Requisitos de Software

* + 1. **Referencias**

Estándar IEEE830

* + 1. **Visión General del Documento**

Este documento consta de tres secciones:

En la primera sección se realiza una introducción al mismo y se proporciona una visión general de la especificación de recursos del sistema.

En la segunda sección del documento se realiza una descripción general del sistema, con el fin de conocer las principales funciones que este debe realizar.

En la tercera sección de documento es donde se definen detalladamente los requisitos que debe satisfacer el sistema.

* 1. **Descripción General**

Implementar un sistema de gestión para el control de calificaciones, para la automatización de los procesos de gestión académica

Evaluar el programa instalado para el registro de notas, para verificar que cumpla con las expectativas propuestas.

* + 1. **Perspectiva del Producto**

Facilitar a los profesores y docentes.

Crear un sistema que tenga interfaces amigables para facilitar la interacción del usuario con el programa.

* + 1. **Funciones del producto**

Registrar de materias

Registrar de estudiantes

Registrar las notas de cada estudiante

Visualizar notas de los estudiantes con su correspondiente materia

* + 1. **Características de los usuarios**

|  |  |
| --- | --- |
| Tipo de Usuario | Docente |
| Formación | Educador |
| Actividad | Puede registrar materias, estudiantes, y las notas de sus estudiantes. |

* + 1. **Restricciones**

Interfaz amigable

Lenguaje de programación: JAVA

El sistema se diseñará según un modelo de cliente

El programa tendrá un diseño e implementación sencilla.

* + 1. **Suposiciones y Dependencias**

Se asume que los requisitos aquí descritos pueden modificarse de acuerdo a los requerimientos.

Los equipos en los que se va ejecutar el programa deben cumplir los requisitos antes indicados para garantizar una ejecución correcta de la misma.

* + 1. **Requisitos Futuros**

Hacer que el programa sea multiplataforma.

Consultas requeridas por los usuarios.

Poder intercambiar información entre instituciones.

Permitir que el estudiante tenga acceso a su información de notas en sus respectivas materias.

* 1. **Requisitos Específicos**

A continuación, se mostrará los requerimientos funcionales que tendrá el programa:

Requerimientos Funcionales

Constará básicamente de 3 módulos que engloban todo:

Registrar Materia

En esta parte se hará el registro de las materias que el profesor o docente tiene.

Registrar Estudiantes

El docente podrá registrar a sus estudiantes que tienen de las diferentes materias.

Registrar Nota

El docente podrá registrar las notas de sus estudiantes de acuerdo a sus respectivas materias que tiene registrado.

Los requerimientos mencionados poseen las características de:

Correcto: todos los datos ingresados tienen restricciones de poner códigos repetidos.

No ambiguo: no hay ambigüedad en los datos.

Consistente: los datos quedan consistentes, gracias a las restricciones de entrada.

Verificable: se pueden hacer pruebas.

* + 1. **Interfaces Externas**

Usando una tarjeta de video mejora la interfaz y sebera mucho mejor la imagen y más fluido al momento de usarlo

Una recomendación instalar java para mejor comunicación entre programas.

* + 1. **Funciones**

El programa constará de las siguientes funciones:

* Previo a todo, el usuario podrá registrar a los estudiantes a sus diferentes materias.
* Se podrá asignar nota a cada estudiante de acuerdo a las diferentes materias que se tiene registrado.
* Se podrá observar un listado de los estudiantes con las diferentes materias que tiene registrado cada uno de los estudiantes, y con su respectiva nota asignada.
  + 1. **Requisitos de Rendimiento**

Windows 10

RAM: 128 MB

Espacio en disco: 124 MB para JRE; 2 MB para Java Update

Procesador: Mínimo Pentium 2 a 266 MHz

* + 1. **Restricciones de Diseño**

Algunas son que el usuario no cuente con las características necesarias que requiere el programa para poder funcionar como tiene que ser.

* + 1. **Atributos del Sistema**

Se puede ocultar el programa para que no puedan alterar las notas y solo el profesor sabe dónde se encuentra para poder usarlo con seguridad.

* + 1. **Otros Requisitos**

Tener cualquier navegador de internet para realizar actualizaciones del programa y acceso a internet.

1. **ANÁLISIS**

A continuación, se analizará y se especificará a detalle cada requerimiento mencionado anteriormente.

|  |  |
| --- | --- |
| **REQUERIMIENTOS FUNCIONALES** | |
| **ACCION** | REGISTRAR MATERIA |
| **DESCRIPCION** | En esta parte se hará el registro de las materias que el profesor o docente tiene. |
| **ENTRADA** | 1. Ingresar nombre de la materia |
|  | 2. Ingresa el código del estudiante |
|  |  |
| **SALIDA** | Insertar al listado |

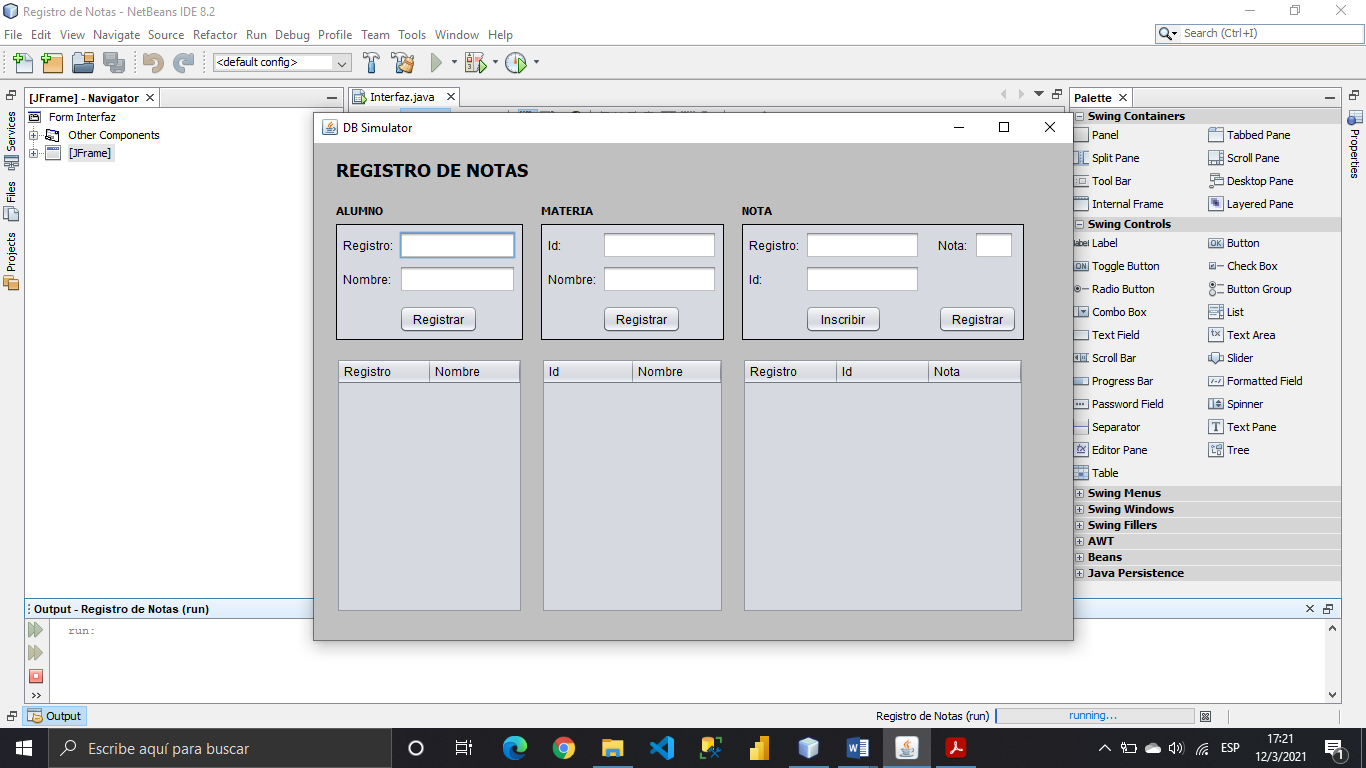
|  |  |
| --- | --- |
| **REQUERIMIENTOS FUNCIONALES** | |
| **ACCION** | REGISTRAR ESTUDIANTE |
| **DESCRIPCION** | El docente podrá registrar a sus estudiantes que tienen de las diferentes materias. |
| **ENTRADA** | 1. Ingresar nombre del estudiante |
|  | 2. Ingresar el código del estudiante |
|  |  |
| **SALIDA** | Insertar al listado |

|  |  |
| --- | --- |
| **REQUERIMIENTOS FUNCIONALES** | |
| **ACCION** | REGISTRAR NOTA |
| **DESCRIPCION** | El docente podrá registrar las notas de sus estudiantes de acuerdo a sus respectivas materias que tiene registrado. |
| **ENTRADA** | 1. Seleccionar del listado |
|  | 2. Ingresar la nota asignada |
|  |  |
| **SALIDA** | Modificar el listado |

1. **DISEÑO**

En el diseño modelamos el sistema y buscamos incluir todos los requisitos funcionales y otras restricciones que se le suponen.

A continuación, vemos el diseño de la interfaz que es prácticamente donde el usuario interactúa con nuestro sistema, ingresando datos y también podrá ver los datos ingresados.



1. **DESARROLLO**

En esta parte se observa toda la codificación de nuestro programa implementado.

**Interfaz**

public class Interfaz extends javax.swing.JFrame {

ArrayList<TAlumno> Alumno = new ArrayList<>();

ArrayList<TMateria> Materia = new ArrayList<>();

ArrayList<TNota> Nota = new ArrayList<>();

//copia del arraylist Nota

ArrayList<TNota2> Nota2 = new ArrayList<>();

public Interfaz() {

initComponents();

setLocationRelativeTo(null);

this.getContentPane().setBackground(Color.lightGray);

}

private void jButton\_RegistrarAlumnoActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (jTextField\_RegistroAlumno.getText().equals("") || jTextField\_NombreAlumno.getText().equals("")) {

JOptionPane.showMessageDialog(null, "No puede haber campos vacíos");

} else {

boolean val = true;

if (Alumno.size()>=1) {

val = ValidarAlumno();

}

if (val == false) {

JOptionPane.showMessageDialog(null, "Ya existe un alumno con ese registro");

} else {

TAlumno TAlumno1 = new TAlumno(Integer.parseInt(jTextField\_RegistroAlumno.getText()),

jTextField\_NombreAlumno.getText());

Alumno.add(TAlumno1);

MostrarAlumno();

jTextField\_RegistroAlumno.setText("");

jTextField\_NombreAlumno.setText("");

}

}

}

private void jButton\_RegistrarMateriaActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (jTextField\_IdMateria.getText().equals("") || jTextField\_NombreMateria.getText().equals("")) {

JOptionPane.showMessageDialog(null, "No puede haber campos vacíos");

} else {

boolean val = true;

if (Materia.size()>=1) {

val = ValidarMateria();

}

if (val == false) {

JOptionPane.showMessageDialog(null, "Ya existe una materia con ese id");

} else {

TMateria TMateria1 = new TMateria(Integer.parseInt(jTextField\_IdMateria.getText()),

jTextField\_NombreMateria.getText());

Materia.add(TMateria1);

MostrarMateria();

jTextField\_IdMateria.setText("");

jTextField\_NombreMateria.setText("");

}

}

}

private void jButton\_RegistrarNotaActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (jTextField\_RegistroNota.getText().equals("") || jTextField\_IdNota.getText().equals("") ||

jTextField\_Nota.getText().equals("")) {

JOptionPane.showMessageDialog(null, "No puede haber campos vacíos");

} else {

if (ExisteAlumno(Integer.parseInt(jTextField\_RegistroNota.getText())) &&

ExisteMateria(Integer.parseInt(jTextField\_IdNota.getText()))) {

int pos = BuscarInscripcion();

if (pos >= 0) {

if (TieneNota(pos) == true) {

JOptionPane.showMessageDialog(null, "El estudiante ya tiene nota en esa materia");

} else {

boolean val = ValidarNota(Integer.parseInt(jTextField\_Nota.getText()));

if (val == false) {

JOptionPane.showMessageDialog(null, "Nota inválida");

} else {

Nota.get(pos).setNota(Integer.parseInt(jTextField\_Nota.getText()));

Nota2.get(pos).setNota(Integer.parseInt(jTextField\_Nota.getText()));

MostrarNota();

jTextField\_RegistroNota.setText("");

jTextField\_IdNota.setText("");

jTextField\_Nota.setText("");

}

}

} else {

JOptionPane.showMessageDialog(null, "El estudiante no esta registrado en esa materia");

}

} else {

JOptionPane.showMessageDialog(null, "Datos no encontrados");

}

}

}

private void jButton\_InscribirAlumnoActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (jTextField\_RegistroNota.getText().equals("") || jTextField\_IdNota.getText().equals("")) {

JOptionPane.showMessageDialog(null, "No puede haber campos vacíos");

} else {

if (ExisteAlumno(Integer.parseInt(jTextField\_RegistroNota.getText())) &&

ExisteMateria(Integer.parseInt(jTextField\_IdNota.getText()))) {

boolean val = true;

if (Nota.size()>=1)

val = ValidarInscripcion();

if (val == false) {

JOptionPane.showMessageDialog(null, "El estudiante ya esta inscrito en esa materia");

} else {

int pos\_a = EncontrarAlumno(Integer.parseInt(jTextField\_RegistroNota.getText()));

int pos\_m = EncontrarMateria(Integer.parseInt(jTextField\_IdNota.getText()));

TNota TNota1 = new TNota((Alumno.get(pos\_a)),(Materia.get(pos\_m)),-1);

TNota2 TNotaM = new TNota2((Alumno.get(pos\_a).getRegistro()),(Materia.get(pos\_m).getId()),-1);

Nota.add(TNota1);

Nota2.add(TNotaM);

MostrarNota();

jTextField\_RegistroNota.setText("");

jTextField\_IdNota.setText("");

}

} else {

JOptionPane.showMessageDialog(null, "Datos no encontrados");

}

}

}

private void jTableNotaMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

int select = jTableNota.rowAtPoint(evt.getPoint());

jTextField\_RegistroNota.setText(String.valueOf(jTableNota.getValueAt(select, 0)));

jTextField\_IdNota.setText(String.valueOf(jTableNota.getValueAt(select, 1)));

}

public boolean ValidarInscripcion() {

for (int i = 0; i < Nota.size(); i++) {

if (((Integer.parseInt(jTextField\_RegistroNota.getText())==Nota2.get(i).getRegistro())) &&

((Integer.parseInt(jTextField\_IdNota.getText())==Nota2.get(i).getId())))

return false;

}

return true;

}

public int BuscarInscripcion() {

int s = -1;

for (int i = 0; i < Nota.size(); i++) {

if (((Integer.parseInt(jTextField\_RegistroNota.getText())==Nota2.get(i).getRegistro())) &&

((Integer.parseInt(jTextField\_IdNota.getText())==Nota2.get(i).getId())))

return s = i;

}

return s;

}

public boolean TieneNota(int p) {

return (Nota.get(p).getNota())>=0;

}

public boolean ValidarNota(int n) {

return (n>=0) && (n<=100);

}

public boolean ExisteAlumno(int reg) {

for (int i = 0; i < Alumno.size(); i++) {

if (reg == Alumno.get(i).getRegistro()) {

return true;

}

}

return false;

}

public boolean ExisteMateria(int id) {

for (int i = 0; i < Materia.size(); i++) {

if (id == Materia.get(i).getId()) {

return true;

}

}

return false;

}

public int EncontrarAlumno(int reg) {

for (int i = 0; i < Alumno.size(); i++) {

if (reg == Alumno.get(i).getRegistro()) {

return i;

}

}

return -1;

}

public int EncontrarMateria(int id) {

for (int i = 0; i < Materia.size(); i++) {

if (id == Materia.get(i).getId()) {

return i;

}

}

return -1;

}

public boolean ValidarAlumno() {

for (int i = 0; i < Alumno.size(); i++) {

if (Integer.parseInt(jTextField\_RegistroAlumno.getText()) == Alumno.get(i).getRegistro()) {

return false;

}

}

return true;

}

public boolean ValidarMateria() {

for (int i = 0; i < Materia.size(); i++) {

if (Integer.parseInt(jTextField\_IdMateria.getText()) == Materia.get(i).getId()) {

return false;

}

}

return true;

}

public void MostrarAlumno() {

String matriz1 [][] = new String[Alumno.size()][2];

for (int i = 0; i < Alumno.size(); i++) {

matriz1[i][0] = Integer.toString(Alumno.get(i).getRegistro());

matriz1[i][1] = Alumno.get(i).getNombre();

}

jTableAlumno.setModel(new javax.swing.table.DefaultTableModel(

matriz1,

new String [] {

"Registro", "Nombre"

}

));

}

public void MostrarMateria() {

String matriz2 [][] = new String[Materia.size()][2];

for (int i = 0; i < Materia.size(); i++) {

matriz2[i][0] = Integer.toString(Materia.get(i).getId());

matriz2[i][1] = Materia.get(i).getNombre();

}

jTableMateria.setModel(new javax.swing.table.DefaultTableModel(

matriz2,

new String [] {

"Id", "Nombre"

}

));

}

public void MostrarNota() {

String matriz3 [][] = new String[Nota.size()][3];

for (int i = 0; i < Nota.size(); i++) {

matriz3[i][0] = Integer.toString(Nota2.get(i).getRegistro());

matriz3[i][1] = Integer.toString(Nota2.get(i).getId());

matriz3[i][2] = Integer.toString(Nota2.get(i).getNota());

}

jTableNota.setModel(new javax.swing.table.DefaultTableModel(

matriz3,

new String [] {

"Registro", "Id", "Nota"

}

));

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Interfaz.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Interfaz().setVisible(true);

}

});

}

**Clase Alumno**

package RegistroDeNotas;

public class TAlumno {

int registro;

String Nombre;

public TAlumno(int registro, String Nombre) {

this.registro = registro;

this.Nombre = Nombre;

}

public int getRegistro() {

return registro;

}

public void setRegistro(int registro) {

this.registro = registro;

}

public String getNombre() {

return Nombre;

}

public void setNombre(String Nombre) {

this.Nombre = Nombre;

}

}

**Clase Materia**

package RegistroDeNotas;

public class TMateria {

int id;

String Nombre;

public TMateria(int id, String Nombre) {

this.id = id;

this.Nombre = Nombre;

}

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;

}

}

**Clase Nota**

package RegistroDeNotas;

public class TNota {

TAlumno Registro;

TMateria Id;

int nota;

public TNota(TAlumno Registro, TMateria Id, int nota) {

this.Registro = Registro;

this.Id = Id;

this.nota = nota;

}

public TAlumno getRegistro() {

return Registro;

}

public void setRegistro(TAlumno Registro) {

this.Registro = Registro;

}

public TMateria getId() {

return Id;

}

public void setId(TMateria Id) {

this.Id = Id;

}

public int getNota() {

return nota;

}

public void setNota(int nota) {

this.nota = nota;

}

}

**Clase Nota2**

package RegistroDeNotas;

public class TNota2 {

int registro;

int id;

int nota;

public TNota2(int registro, int id, int nota) {

this.registro = registro;

this.id = id;

this.nota = nota;

}

public int getRegistro() {

return registro;

}

public void setRegistro(int registro) {

this.registro = registro;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getNota() {

return nota;

}

public void setNota(int nota) {

this.nota = nota;

}

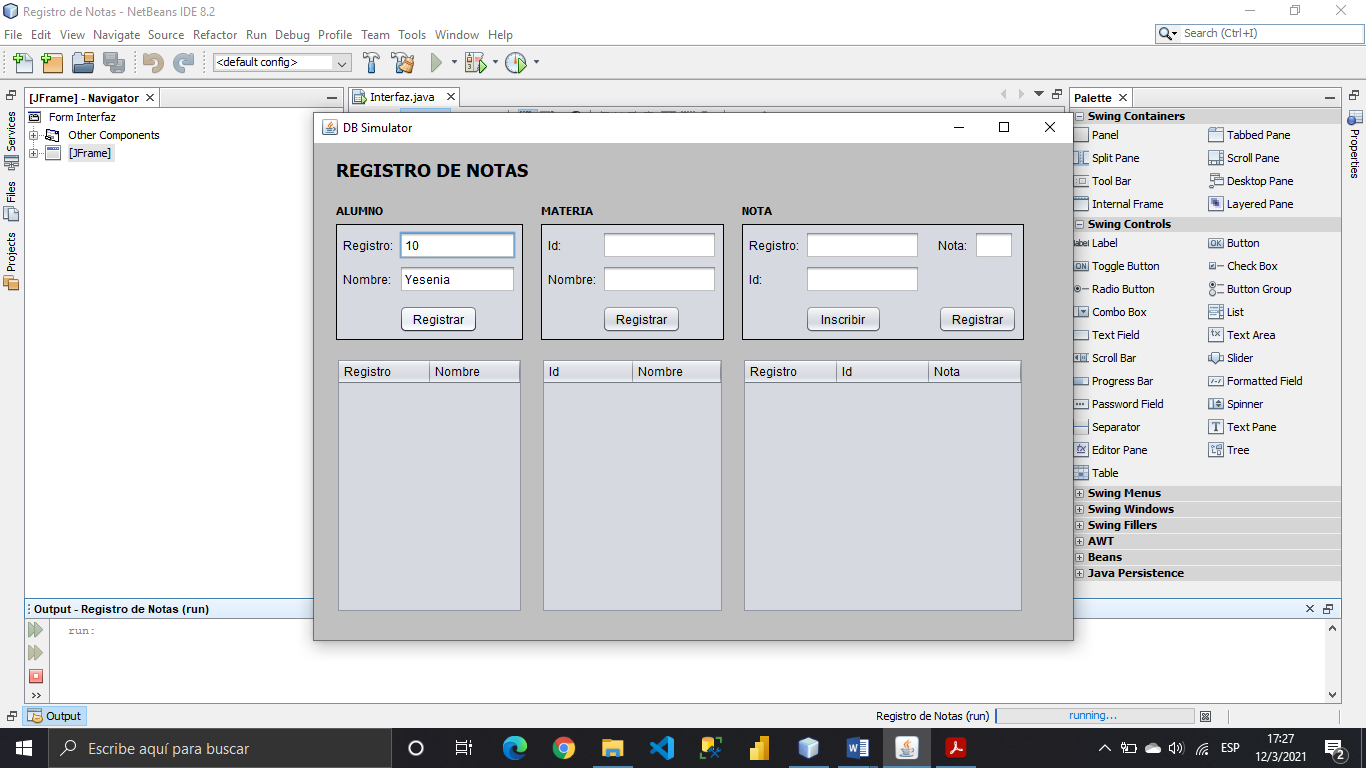
}

1. **PRUEBAS**

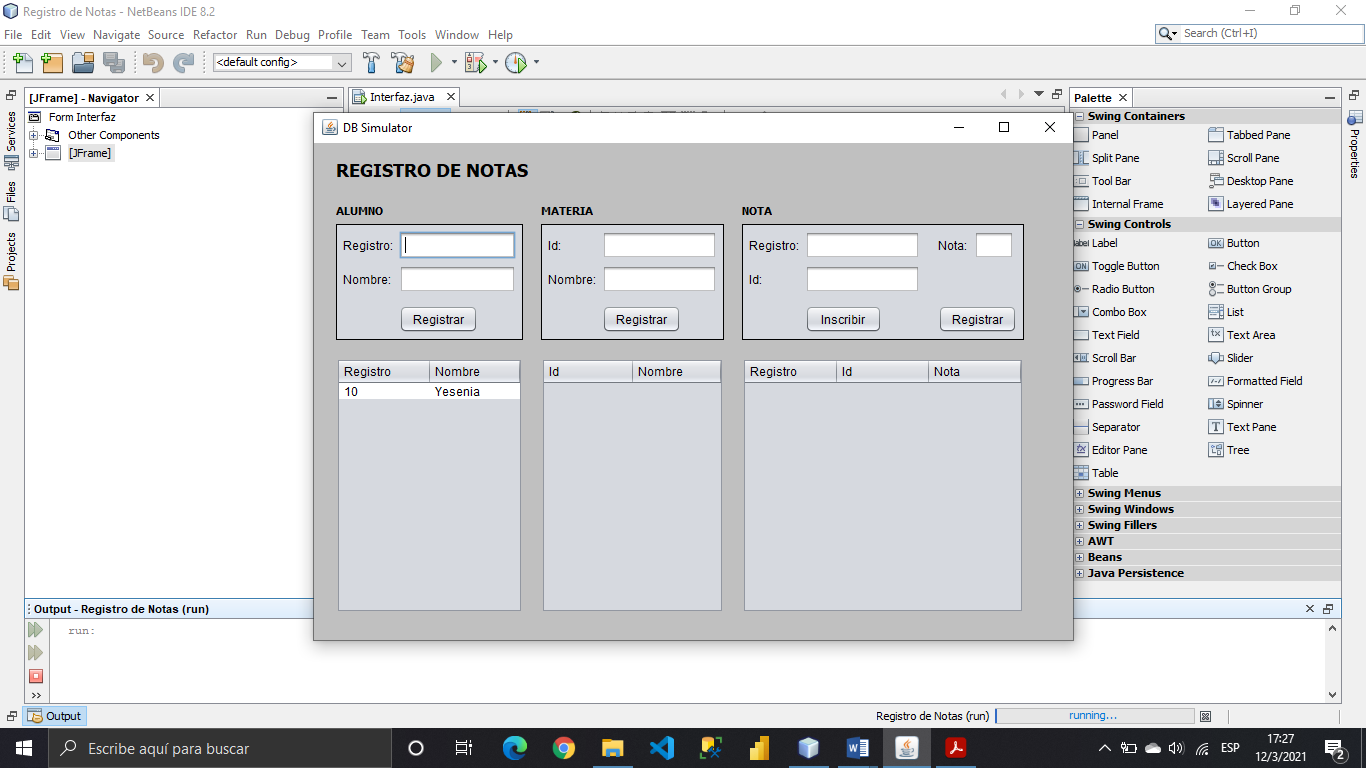
Se procede a la revisión respectiva de cada uno de los módulos de la aplicación, consiste en realizar prueba de validación y verificación a las acciones que realiza los diferentes módulos de la aplicación en donde evidenciaremos su correcto funcionamiento.

La verificación y validación es el nombre que se da a los procesos de comprobación y análisis que aseguran que el software desarrollado este acorde a las especificaciones y cumple las necesidades de los clientes.

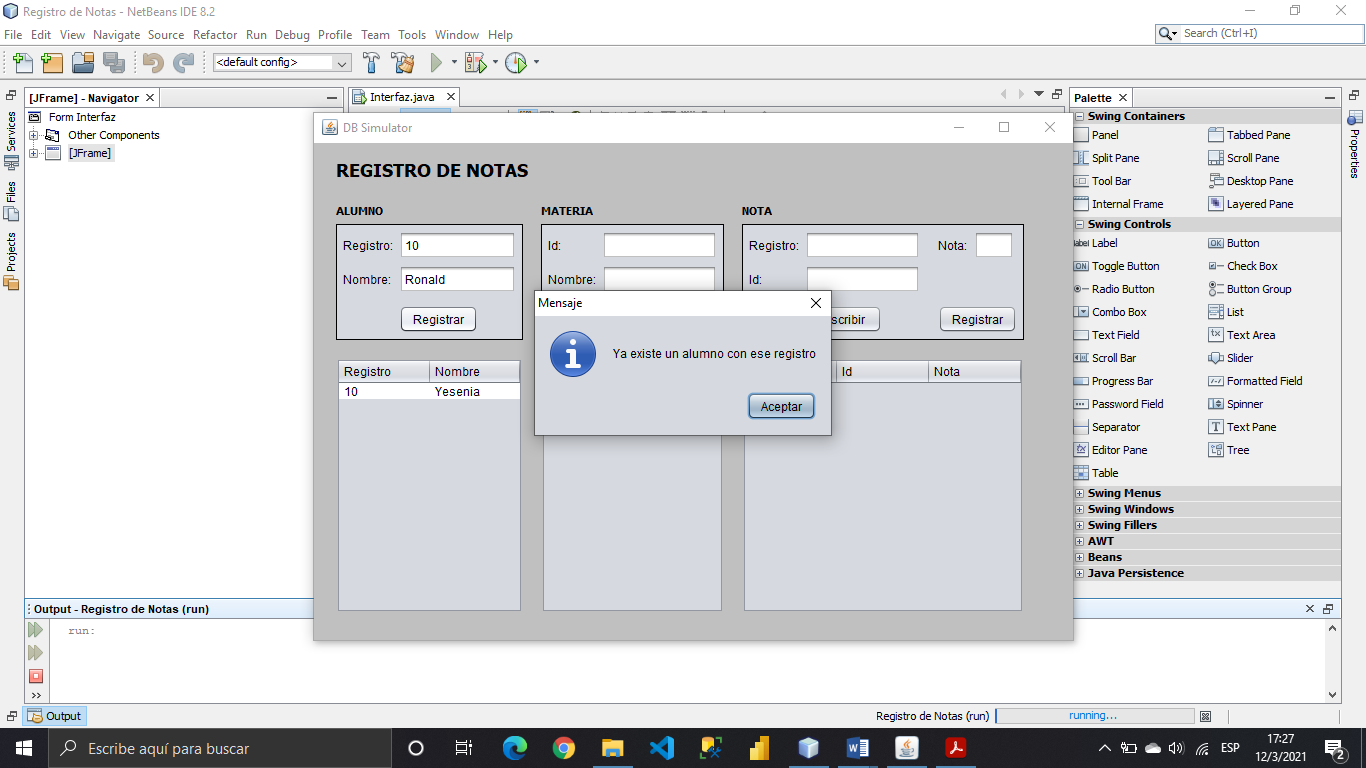
Esta es la vista general



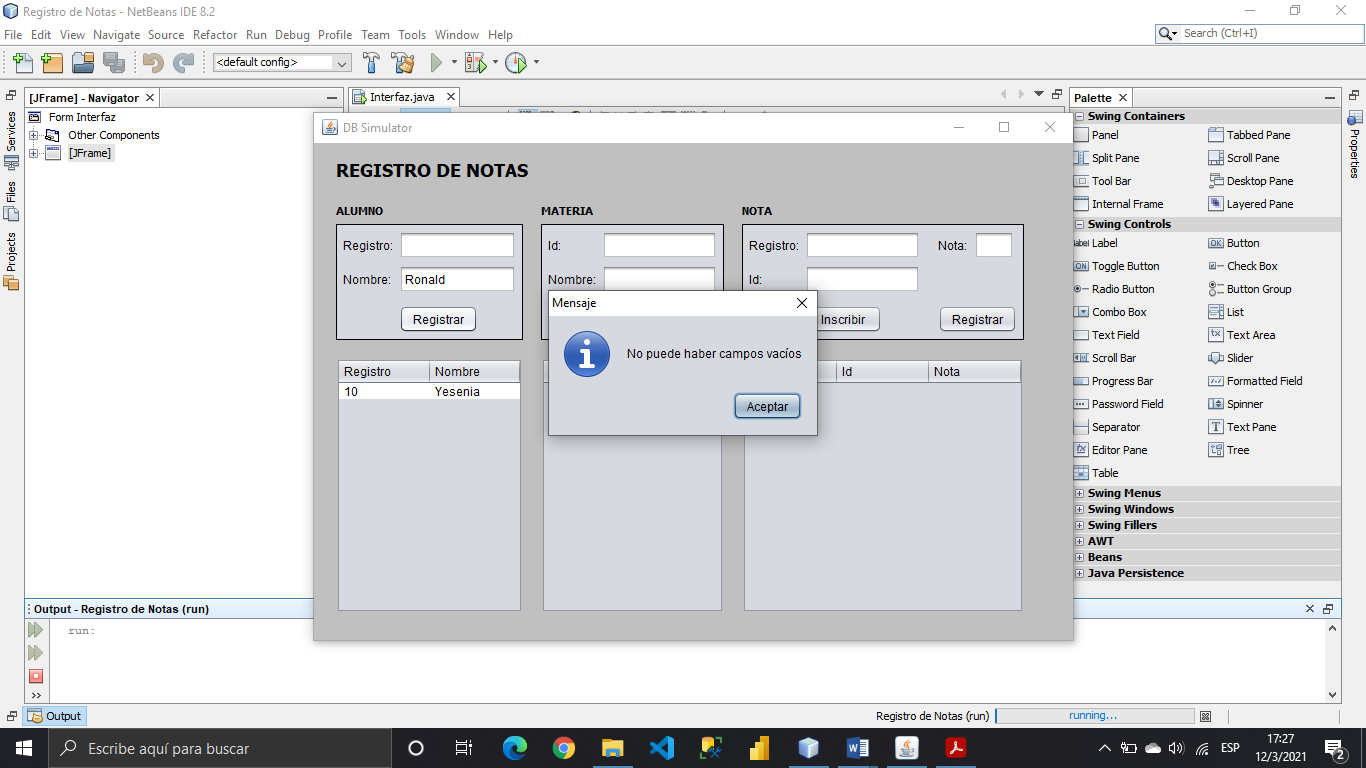
Insertamos datos del alumno



Podemos ver que, si queremos insertar otro alumno con el mismo registro, no nos deja y eso hace que nuestros datos sean consistentes.



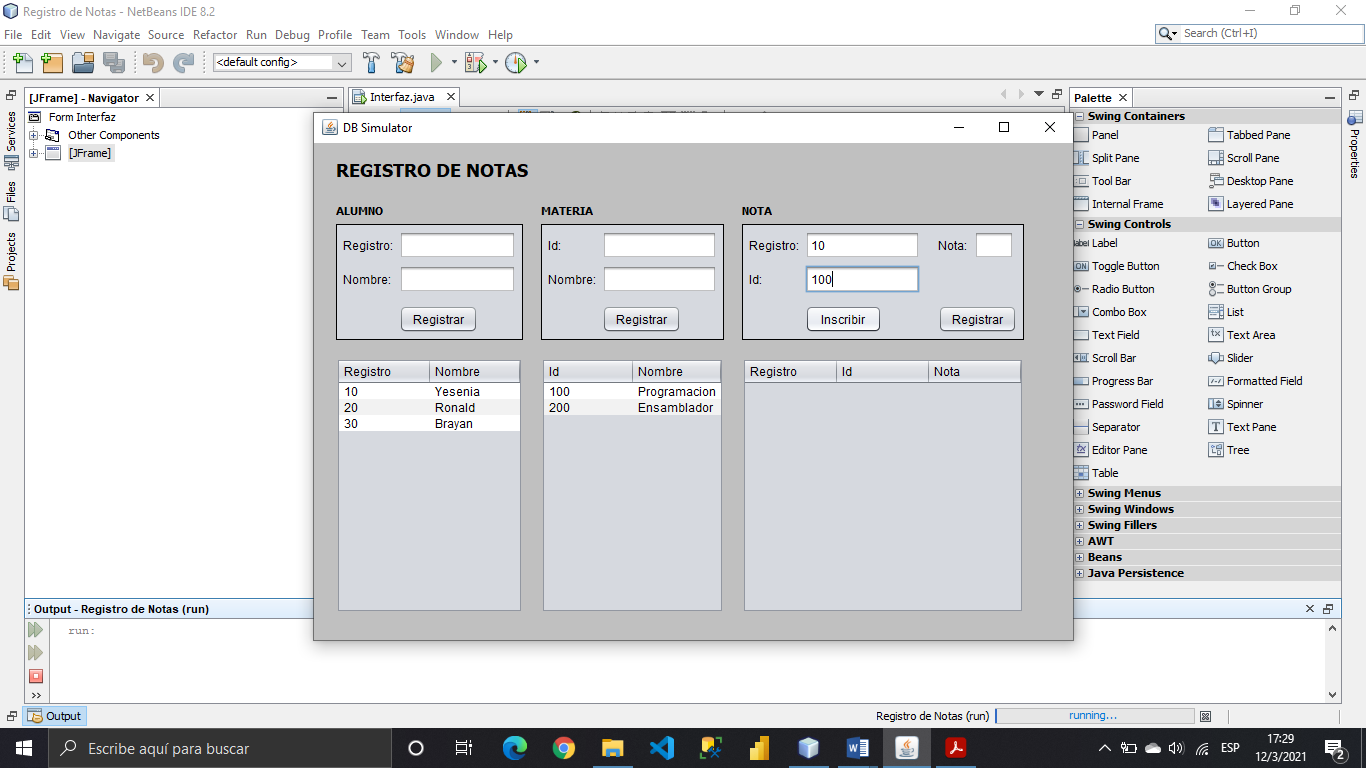
También vemos que restringe los campos vacíos



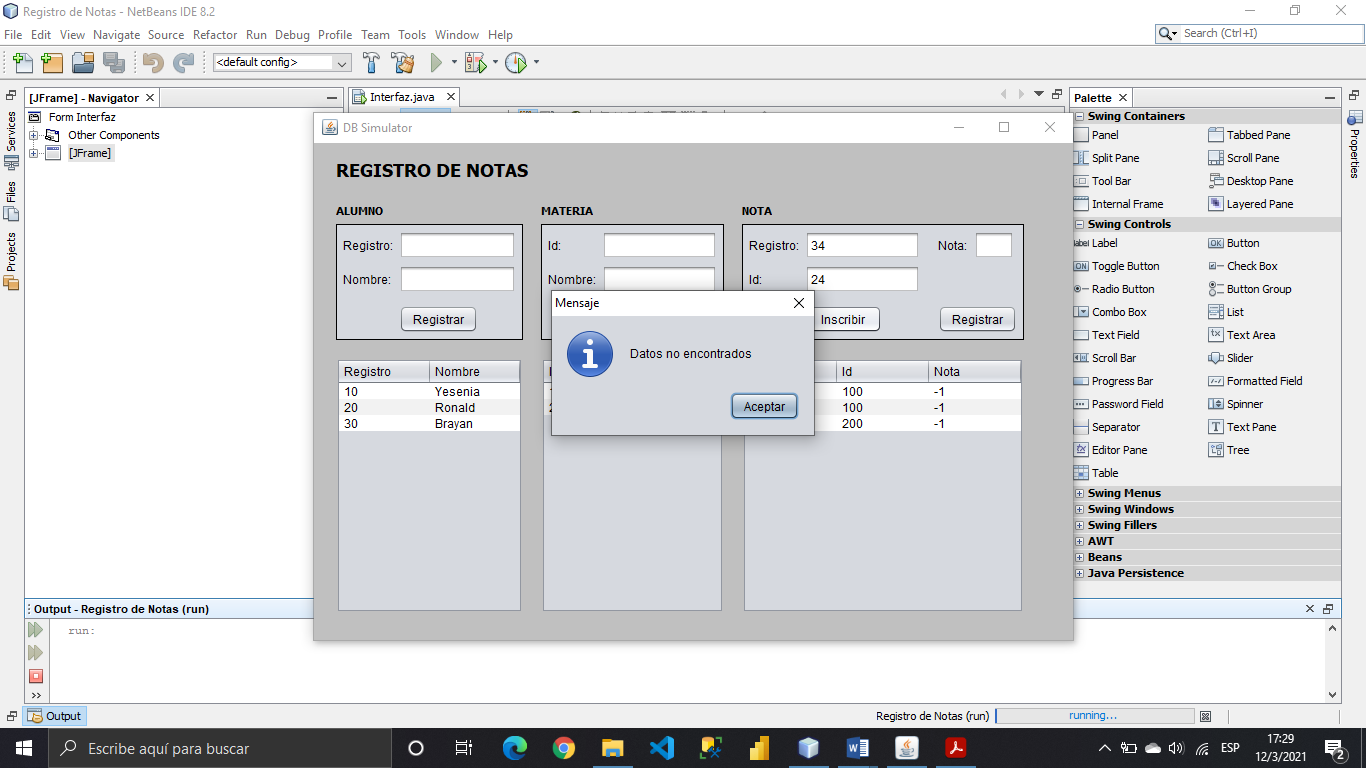
Y asi mismo ingresaremos los datos de materia, que tiene las mismas características de alumno, las mismas restricciones.

Luego como ya tenemos los datos de alumno y materia, procedemos a previamente inscribir al alumno a la materia correspondiente.

Como vemos a continuación.

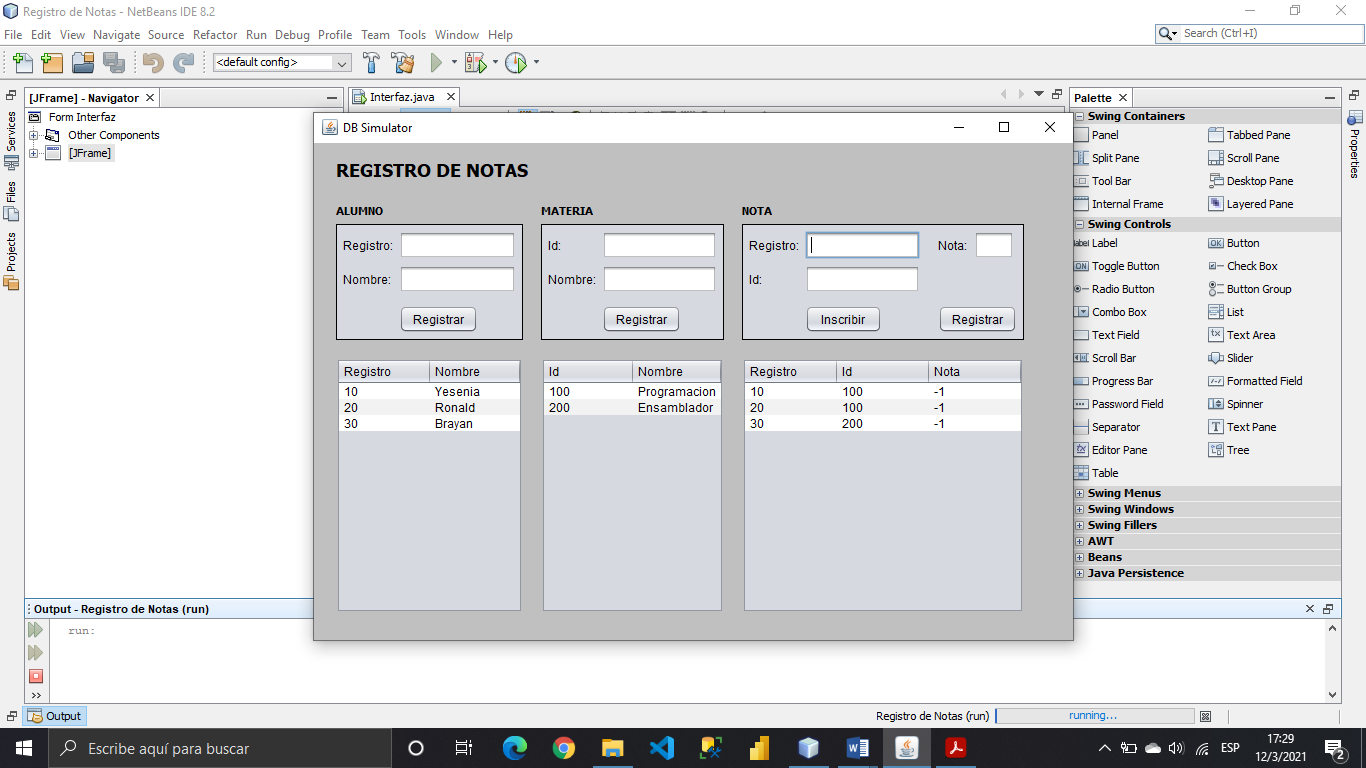


Tambien vemos que esta restrigido, para que el usuario no ponga cualquier dato.



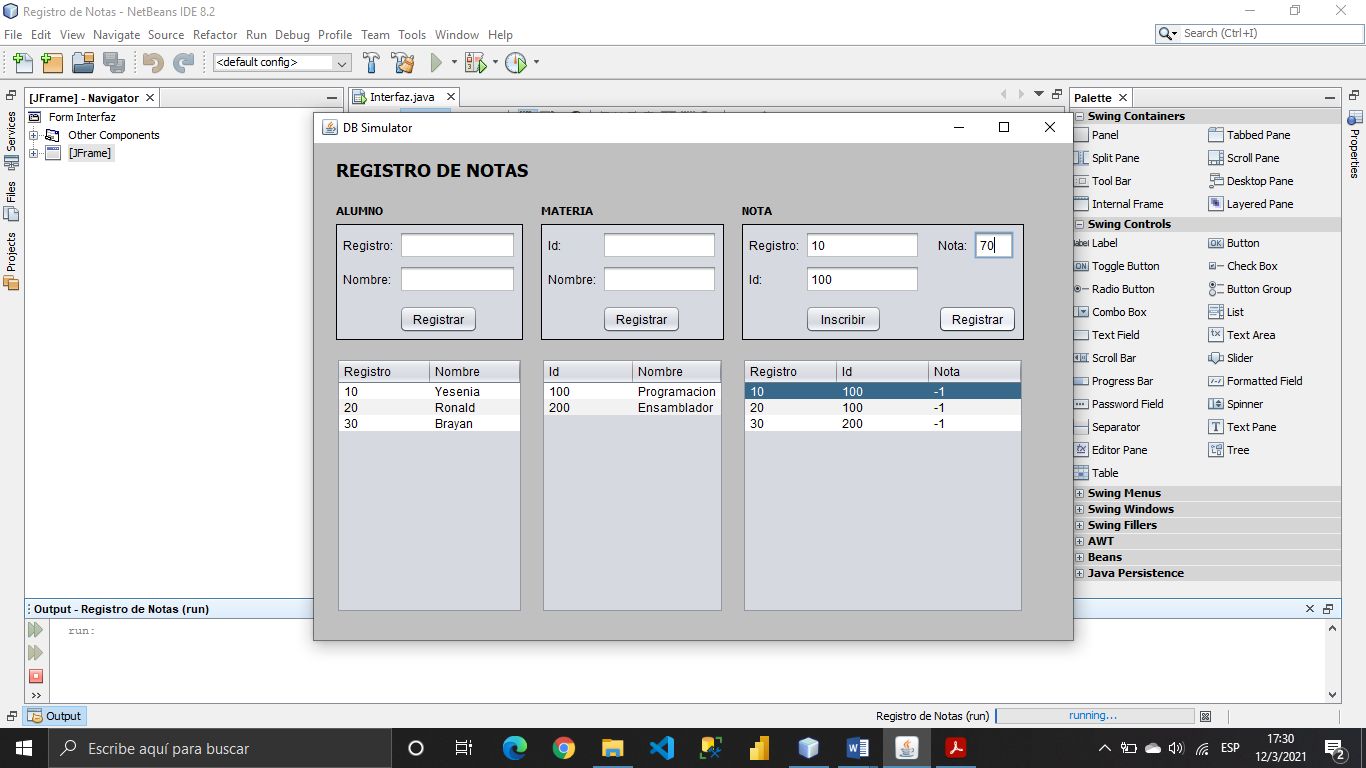
Aquí vemos que ya todos los datos de prueba están insertados

Y se puede observar que, al inscribir al alumno a una materia, la nota se pone en -1 y eso indica que el alumno aún no tiene una nota asignada en dicha materia

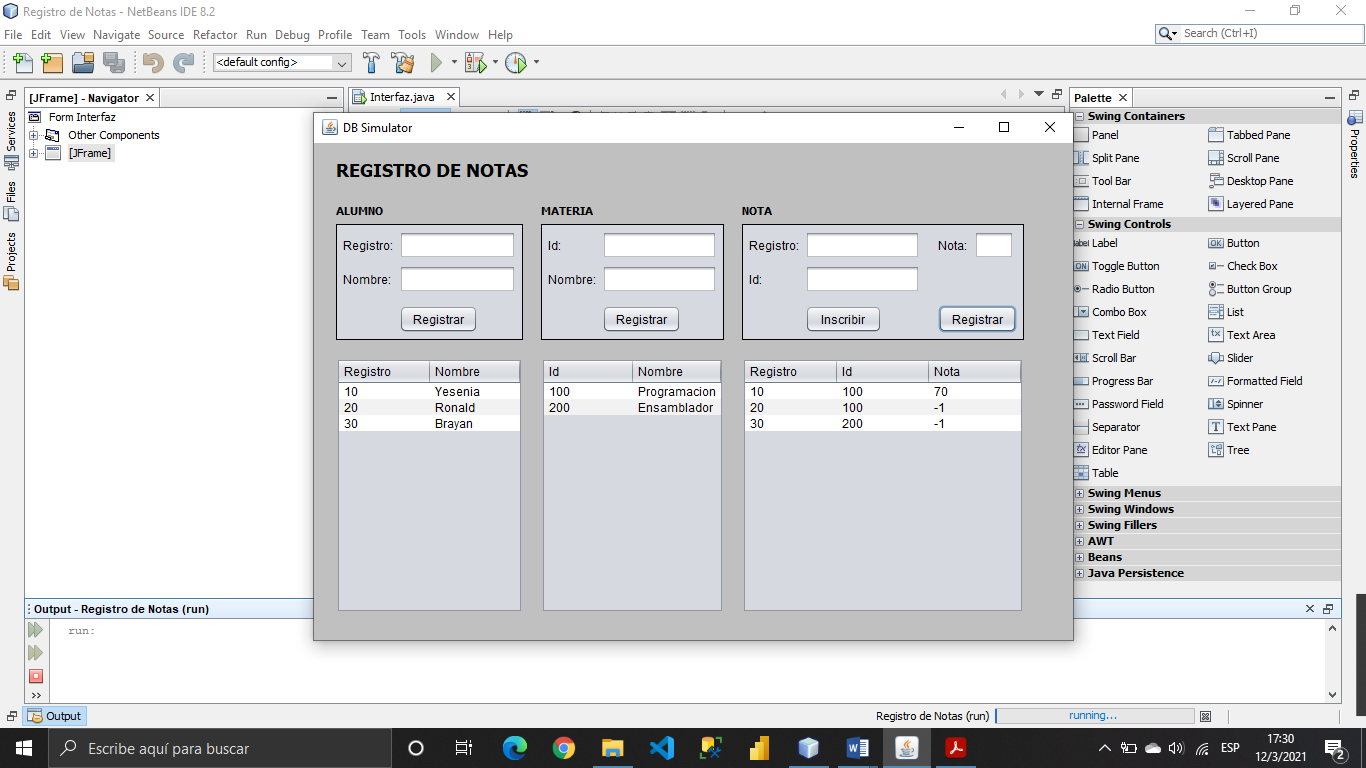


Luego, seleccionamos la fila a modificar, para poder asignar nota al alumno a su correspondiente materia.

Seleccionamos la fila deseada y automáticamente se llenan los datos para solo asignar la nota.



Y vemos que ya la nota se le fue asignada al alumno en su respectiva materia



Y ya finalmente vemos todos los datos ingresados, que son correctos y consistentes.

