



SISTEMA ACADÉMICO DE MATRÍCULA DE UN COLEGIO

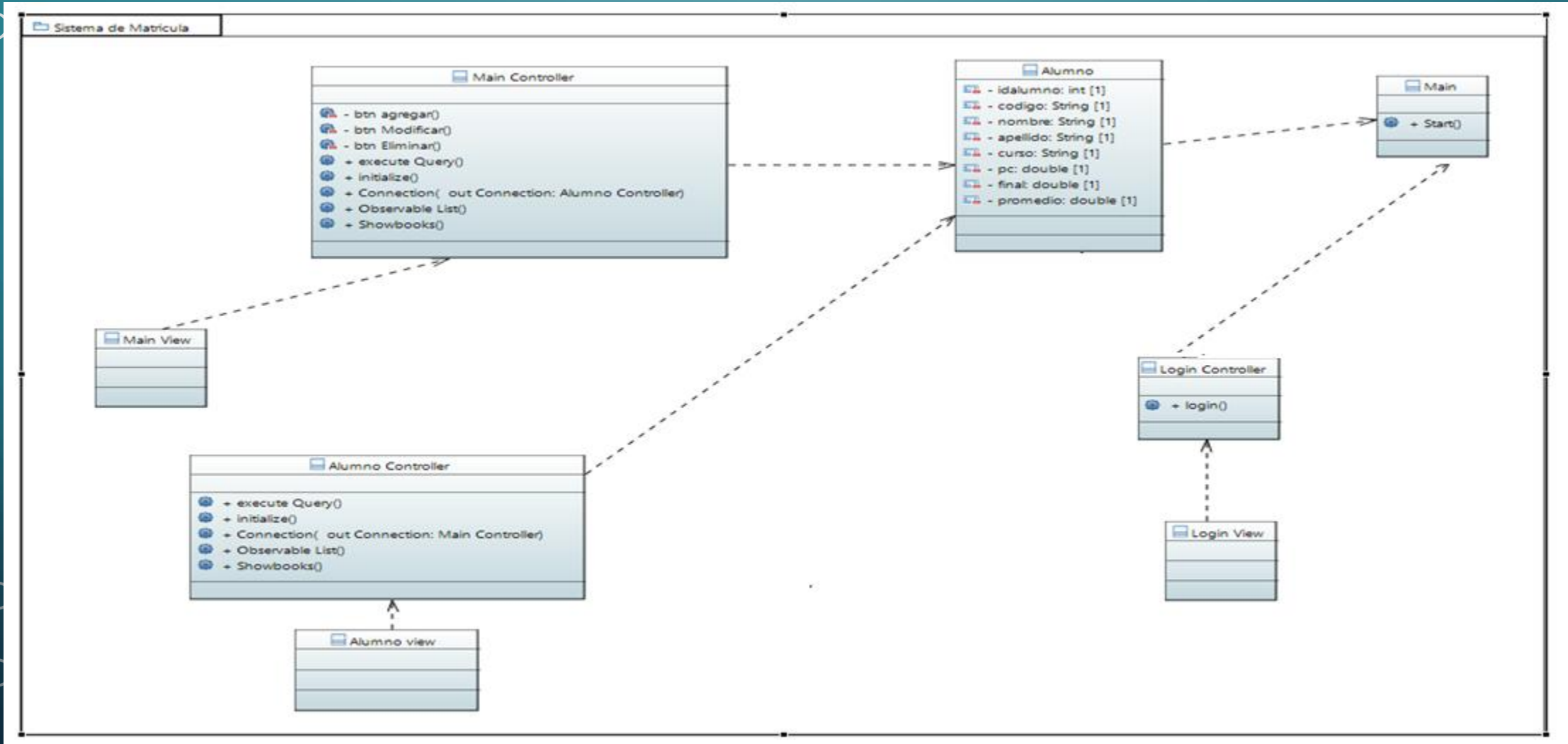
➤ CABELLO NIETO, JOEL MAXIMILIANO	18190089	5 PUNTOS
➤ CHAUCA PRINQUE, JHONNY JUNIOR	18190294	5 PUNTOS
➤ ECHANDIA RAMOS CARLOS JOEL	18190302	5 PUNTOS
➤ LEDESMA MENDOZA LINO JOAQUIN	18190307	5 PUNTOS
➤ TORIBIO CHAVEZ JULIO CESAR	18190344	5 PUNTOS
➤ VILCA RÍOS OSCAR PAOLO	18190323	5 PUNTOS
➤ ZUÑIGA SALAS FRANK	18190325	5 PUNTOS

MOTIVACIÓN Y PROBLEMA

Actualmente casi todos los colegios estatales aún siguen usando un sistema de matrícula presencial que ocasiona que todos los años se forman inmensas colas con la finalidad de obtener una vacante , por eso vamos a desarrollar un software que haga que todo el papeleo que conlleva una matrícula sea mas rápido y fácil.

Lo que nos motivó a desarrollar un sistema que ofrezca de manera eficiente a los padres de familia una manera de simplificar, agilizar y mejorar el proceso de matrícula para sus menores hijos. A su vez se busca brindar de forma continua las notas sobre el menor hijo.

DIAGRAMA DE CLASES



JAVA FX

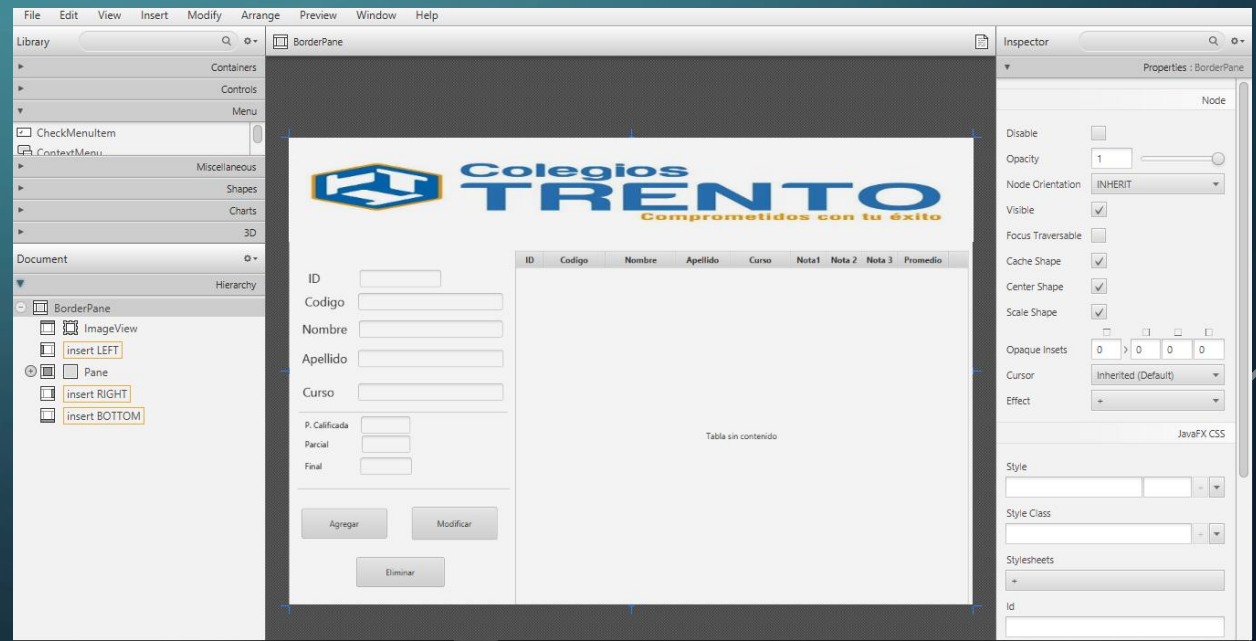
- Para tener un mejor desarrollo del programa hemos usado SceneBuilder para diseñar la interfaz grafica
- Java FX maneja archivos con extensión FXML
- Además al momento de inicializar tenemos que nombrar los componentes de Java FX utilizados

```
@FXML
private PasswordField txtPass;

@FXML
private TextField txtUser;

@FXML
private Label lblStatus;

@FXML
private TableView<Alumno> TableView;
```



CONTENIDOS USADOS

- Validación de usuarios con el uso de condicionales , para permitir el acceso a una Stage (ventana) a través de la clase primaryStage dependiendo si se quiere ingresar como Alumno o Profesor
- Para sacar de un Stage a otro se usa la clase FXMLLoader que carga el archivo XML correspondiente (Ln. 36 y 45)

```
32     if (txtUser.getText().equals("Profesor") && txtPass.getText().equals("123")) {
33
34         lblStatus.setText("Acceso permitido");
35         Stage primaryStage = new Stage();
36         Parent root = FXMLLoader.load(getClass().getResource("/application/Main.fxml"));
37         Scene scene = new Scene(root,1006,697);
38         primaryStage.setScene(scene);
39         primaryStage.show();
40
41     }else if (txtUser.getText().equals("Alumno") && txtPass.getText().equals("123")){
42
43         lblStatus.setText("Acceso permitido");
44         Stage primaryStage = new Stage();
45         Parent root = FXMLLoader.load(getClass().getResource("/application/AlumnoView.fxml"));
46         Scene scene = new Scene(root,690,471);
47         primaryStage.setScene(scene);
48         primaryStage.show();
49
50     }else{
51         lblStatus.setText("Acceso denegado");
52     }
53 }
```

- Este método nos carga el Login a través del FXML (Ln. 15) mostrando la primera ventana

```
13     public void start(Stage stage) throws Exception {  
14  
15         Parent parent = (Parent) FXMLLoader.load(getClass().getResource("/application/Login1.fxml"));  
16  
17         Scene scene = new Scene(parent);  
18         stage.setScene(scene);  
19         stage.setTitle("Matricula");  
20         stage.show();  
21  
22     }  
23  
24
```


- Tenemos que inicializar los atributos de el clase Alumno ,ya que el programa nos pedirá que lo ingresemos para rellenar la base de datos. Usando constructores y métodos Getter

```
public class Alumno {  
  
    private int idAlumno;  
    private String codigo;  
    private String nombre;  
    private String apellido;  
    private String curso;  
    private double pc;  
    private double parcial;  
    private double final1;  
    private double promedio;  
  
    public Alumno (int idAlumno, String codigo, String nombre, String apellido, String curso,  
        double pc, double parcial, double final1, double promedio) {  
        this.idAlumno = idAlumno;  
        this.codigo = codigo;  
        this.nombre = nombre;  
        this.apellido = apellido;  
        this.curso = curso;  
        this.pc = pc;  
        this.parcial = parcial;  
        this.final1 = final1;  
        this.promedio = promedio;  
    }  
}
```

- Primero importamos la clase DecimalFormat para poder calcular el promedio(ln.89)
- Hacemos la consulta a la base de datos mediante el query(consulta) ,ejecutamos la consulta con el método executeQuery(ln.95)
- Listamos todos los datos hasta ahora a través del showBooks (ln. 96)
- Y por ultimo limpiamos las casillas (ln 97–104)

```
88     private void btnAgregar() {
89         DecimalFormat df = new DecimalFormat("#.00");
90         String query = "insert into registro values("+txtIdAlumno.getText()+", "+txtCodigo.getText()+", "
91             + "'"+txtNombre.getText()+", '"+txtApellido.getText()+", '"+txtCurso.getText()+", "
92             + "'"+txtPC.getText()+", "+txtParcial.getText()+", "+txtFinal.getText()+", "
93             + "'"+ df.format((Double.parseDouble(txtParcial.getText())+Double.parseDouble(txtFinal.getText())
94             +Double.parseDouble(txtPC.getText()))/3)+")";
95         executeQuery(query);
96         showBooks();
97         txtIdAlumno.setText("");
98         txtCodigo.setText("");
99         txtNombre.setText("");
100        txtApellido.setText("");
101        txtCurso.setText("");
102        txtPC.setText("");
103        txtParcial.setText("");
104        txtFinal.setText("");
105    }
```


- Primero importamos la clase DecimalFormat para poder calcular el promedio(ln.109)
- Se hace lo mismo que en lo anterior solo que ya no se limpian las casillas
- Solo varia la sentencia SQL a “UPDATE”(ln. 110)

```
108 private void btnModificar() {
109     DecimalFormat df = new DecimalFormat("#.00");
110     String query = "UPDATE registro SETCodigo='"+txtCodigo.getText()+"',Nombre='"+txtNombre.getText()+"'
111         + ",Apellido='"+txtApellido.getText()+"',Curso='"+txtCurso.getText()+"',PC='"+txtPC.getText()+"',
112         + "Parcial='"+txtParcial.getText()+"',Final"+txtFinal.getText()+"',Promedio="
113         +df.format((Double.parseDouble(txtParcial.getText())+Double.parseDouble(txtFinal.getText())
114         +Double.parseDouble(txtPC.getText()))/3)+"')\n WHERE ID='"+txtIdAlumno.getText()+"'";
115     executeQuery(query);
116     showBooks();
117 }
118 }
```

- Para eliminar solo se necesita el ID del alumno y la función del query ejecuta la consulta en la base de datos eliminando el dato seleccionada

```
121     private void btnEliminar() {  
122         String query = "DELETE FROM registro WHERE ID="+txtIdAlumno.getText()+"";  
123         executeQuery(query);  
124         showBooks();  
125     }
```

- Este método se conecta a la base de datos con la clase Connection y luego con el Try Catch intentan ejecutar una sentencia SQL mediante el “createStatement()” si no logra conectar muestra un error

```
127 public void executeQuery(String query) {  
128     Connection conn = getConnection();  
129     Statement st;  
130     try {  
131         st = conn.createStatement();  
132         st.executeUpdate(query);  
133     } catch (Exception e) {  
134         e.printStackTrace();  
135     }  
136 }  
137
```

- Este método permite la conexión del programa con la base de datos mediante una URL en la que se brinda el host , el nombre del esquema , el usuario y la contraseña
- Si no logra conectar bota otro error

```
143 public Connection getConnection() {  
144     Connection conn;  
145     try {  
146         conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/matricula","root","root");  
147         return conn;  
148     }  
149     catch (Exception e){  
150         e.printStackTrace();  
151         return null;  
152     }  
153 }
```

- Este método carga datos de la DB a la tabla , a través de las clases setCellValueFactory que asigna un valor a cada celda

```
179 public void showBooks() {  
180  
181     ObservableList<Alumno> list = getAlumnoList();  
182  
183     colID.setCellValueFactory(new PropertyValueFactory<Alumno,Integer>("idAlumno"));  
184     colCodigo.setCellValueFactory(new PropertyValueFactory<Alumno,String>("codigo"));  
185     colNombre.setCellValueFactory(new PropertyValueFactory<Alumno,String>("nombre"));  
186     colApellido.setCellValueFactory(new PropertyValueFactory<Alumno,String>("apellido"));  
187     colCurso.setCellValueFactory(new PropertyValueFactory<Alumno,String>("curso"));  
188     colPC.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("pc"));  
189     colParcial.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("parcial"));  
190     colFinal.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("final1"));  
191     colPromedio.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("promedio"));  
192  
193     TableView.setItems(list);  
194 }  
195
```

- Cuando se logea como Alumno el programa solo permite observar las datos guardados en el DB.
- El método showBooks se encargo de ellos mostrando todos los datos que están guardados.

```
101 public void showBooks() {
102
103     ObservableList<Alumno> list = getAlumnoList();
104
105     colID.setCellValueFactory(new PropertyValueFactory<Alumno,Integer>("idAlumno"));
106     colCodigo.setCellValueFactory(new PropertyValueFactory<Alumno,String>("codigo"));
107     colNombre.setCellValueFactory(new PropertyValueFactory<Alumno,String>("nombre"));
108     colApellido.setCellValueFactory(new PropertyValueFactory<Alumno,String>("apellido"));
109     colCurso.setCellValueFactory(new PropertyValueFactory<Alumno,String>("curso"));
110     colPC.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("pc"));
111     colParcial.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("parcial"));
112     colFinal.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("final1"));
113     colPromedio.setCellValueFactory(new PropertyValueFactory<Alumno,Double>("promedio"));
114
115     TableView.setItems(list);
116
117
118 }
```


BASE DE DATOS MYSQL

Nuestra conexión tiene como usuario a root y como puerto el 3306



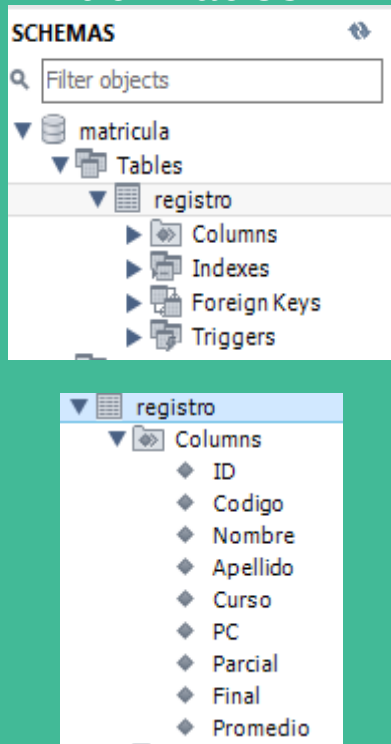
The screenshot shows the 'MySQL Connections' window. It lists a 'Local instance MySQL80'. Below the instance name, it shows the user 'root' and the connection details 'localhost:3306'. A faint elephant logo is visible in the background of the window.

Local instance MySQL80

root

localhost:3306

Esquemas y tablas registro utilizados



The image shows a screenshot of a database management tool interface. The top window, titled 'SCHEMAS', displays a tree view of the database structure. Under the 'matricula' schema, the 'registro' table is selected. The table's properties are shown in a lower pane, listing the following columns: ID, Codigo, Nombre, Apellido, Curso, PC, Parcial, Final, and Promedio. Each column is preceded by a diamond icon, indicating its data type.

SCHEMAS

Filter objects

matricula

Tables

registro

Columns

Indexes

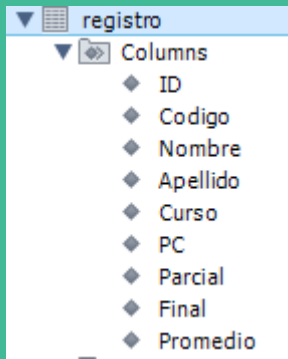
Foreign Keys

Triggers

registro

Columns

- ID
- Codigo
- Nombre
- Apellido
- Curso
- PC
- Parcial
- Final
- Promedio


[illegible][illegible]

Matricula

Login

Contraseña

Ingresar



Colegios
TRENTO
Comprometidos con tu éxito

ID

Codigo

Nombre

Apellido

Curso

P. Calificada

Parcial

Final

Agregar

Modificar

Eliminar

ID	Codigo	Nombre	Apellido	Curso	Nota1	Nota 2	Nota 3	Promedio
Tabla sin contenido								

Notas generales. Ciclo 2019-I

ID	Codigo	Nombre	Apellido	Curso	PC	Parcial	Final	Promedio
Tabla sin contenido								