SISTEMA DE GESTIÓN ACADÉMICA

DESCRIPCIÓN DEL PROYECTO

Sistema de gestión académica desarrollado en Python más Oracle 11g actualizado a Java más Oracle 11 g que permite administrar estudiantes, cursos, docentes y matrículas e incluye procedimientos almacenados, triggers, vistas avanzadas y una interfaz Java amigable en java.

Tablas Principales

a) Estudiante: Datos personales de estudiantes

b) Docente: Información de profesores

c) Curso: Cursos académicos ofrecidos

d) Matricula: Relación estudiantes-cursos

e) Auditoria_Estudiante: Historial de cambios (NUEVO)

FUNCIONALIDADES IMPLEMENTADAS

CRUD Básico

a) Insertar, actualizar, eliminar y consultar estudiantes

b) Gestión de cursos y docentes

c) Sistema de matrículas

Componentes avanzados para la entrega 3

a) 3 procedimientos Almacenados
b) 2Triggers (auditoría + validación)
c) 3 vistas (reportes avanzados)
d) Sistema de Auditoría automático

1) **SCRIPT SQL**
   Las referencias de cada parte se encuentran dentro de las tablas y demás creados en este codigo

```
2) -- ========================================
3) -- SISTEMA DE GESTIÓN ACADÉMICA - ORACLE
4) -- ========================================
5)
```

SECUENCIAS

```
CREATE SEQUENCE seq_estudiante START WITH 1 INCREMENT BY 1 NOCACHE NOCYCLE;
CREATE SEQUENCE seq_curso START WITH 100 INCREMENT BY 1 NOCACHE NOCYCLE;
CREATE SEQUENCE seq_docente START WITH 200 INCREMENT BY 1 NOCACHE NOCYCLE;
CREATE SEQUENCE seq_matricula START WITH 1000 INCREMENT BY 1 NOCACHE NOCYCLE;

```

TABLAS PRINCIPALES

```
CREATE TABLE Estudiante (
    id_estudiante NUMBER PRIMARY KEY,
    nombre VARCHAR2(50) NOT NULL,
    apellido VARCHAR2(50) NOT NULL,
    dni VARCHAR2(10) UNIQUE,
    email VARCHAR2(100)
);

CREATE TABLE Docente (
    id_docente NUMBER PRIMARY KEY,
    nombre VARCHAR2(100) NOT NULL,
    especialidad VARCHAR2(50),
    email VARCHAR2(100)
);

CREATE TABLE Curso (
    id_curso NUMBER PRIMARY KEY,
    nombre VARCHAR2(100) NOT NULL,
    duracion_semanas NUMBER CHECK (duracion_semanas > 8),
    modalidad VARCHAR2(20) CHECK (modalidad IN ('Presencial', 'Online')),
    id_docente NUMBER,
    CONSTRAINT fk_curso_docente FOREIGN KEY (id_docente) REFERENCES Docente(id_docente)
);

CREATE TABLE Matricula (
    id_matricula NUMBER PRIMARY KEY,
    fecha DATE DEFAULT SYSDATE,
    id_estudiante NUMBER,
    id_curso NUMBER,
```

```
    CONSTRAINT fk_mat_est FOREIGN KEY (id_estudiante) REFERENCES Estudiante(id_estudiante),
    CONSTRAINT fk_mat_curso FOREIGN KEY (id_curso) REFERENCES Curso(id_curso)
);

-- DATOS DE PRUEBA
INSERT INTO Estudiante VALUES (seq_estudiante.NEXTVAL, 'Grecia', 'López', '75123456', 'grecia@email.com');
INSERT INTO Estudiante VALUES (seq_estudiante.NEXTVAL, 'Carlos', 'Mendoza', '75234567', 'carlos.mendoza@gmail.com');

INSERT INTO Docente VALUES (seq_docente.NEXTVAL, 'Dr. Ana Ruiz', 'Desarrollo', 'ana.ruiz@gmail.com');
INSERT INTO Docente VALUES (seq_docente.NEXTVAL, 'Ing. Luis Torres', 'Bases de Datos', 'luis.torres.bd@gmail.com');

INSERT INTO Curso VALUES (seq_curso.NEXTVAL, 'Programación en Python', 12, 'Online', 200);
INSERT INTO Curso VALUES (seq_curso.NEXTVAL, 'Bases de Datos', 10, 'Presencial', 201);

INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, 1, 100);
INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, 2, 101);

-- RESTRICCIONES AVANZADAS
ALTER TABLE Estudiante ADD CONSTRAINT ck_email_est CHECK (email LIKE '%@%.%');
ALTER TABLE Estudiante ADD CONSTRAINT ck_dni_est CHECK (LENGTH(dni) = 8 AND REGEXP_LIKE(dni, '^[0-9]+$'));
ALTER TABLE Curso ADD CONSTRAINT ck_modalidad_curso CHECK (modalidad IN ('Presencial', 'Online', 'Híbrido'));
ALTER TABLE Curso ADD CONSTRAINT ck_duracion_curso CHECK (duracion_semanas BETWEEN 1 AND 52);
ALTER TABLE Matricula ADD CONSTRAINT uk_matricula_unica UNIQUE (id_estudiante, id_curso);
ALTER TABLE Docente ADD CONSTRAINT ck_email_doc CHECK (email LIKE '%@%.%');

-- ÍNDICES
CREATE INDEX idx_est_apellido ON Estudiante(apellido);
CREATE INDEX idx_estudiante_email ON Estudiante(email);
```

```
CREATE INDEX idx_curso_docente ON Curso(id_docente);
CREATE INDEX idx_matricula_fecha ON Matricula(fecha);
CREATE INDEX idx_docente_especialidad ON Docente(especialidad);
CREATE INDEX idx_curso_nombre ON Curso(nombre);

-- ========================================
   COMPONENTES AVANZADOS
-- ========================================

-- PROCEDIMIENTOS ALMACENADOS
CREATE OR REPLACE PROCEDURE sp_estudiantes_por_curso (
    p_id_curso IN NUMBER,
    p_cursor OUT SYS_REFCURSOR
) AS BEGIN
    OPEN p_cursor FOR
    SELECT e.id_estudiante, e.nombre, e.apellido, e.email, e.dni
    FROM Estudiante e JOIN Matricula m ON e.id_estudiante = m.id_estud
   iante
    WHERE m.id_curso = p_id_curso;
END;
/

CREATE OR REPLACE PROCEDURE sp_contar_estudiantes_curso (
    p_id_curso IN NUMBER,
    p_total OUT NUMBER
) AS BEGIN
    SELECT COUNT(*) INTO p_total FROM Matricula WHERE id_curso = p_id_
   curso;
END;
/

CREATE OR REPLACE PROCEDURE sp_matricular_estudiante (
    p_id_estudiante IN NUMBER,
    p_id_curso IN NUMBER,
    p_resultado OUT VARCHAR2
) AS
    v_existe_est NUMBER;
    v_existe_cur NUMBER;
     v_ya_matriculado NUMBER;
 BEGIN
     SELECT COUNT(*) INTO v_existe_est FROM Estudiante WHERE id_estudi
   ante = p_id_estudiante;
```

```
    IF v_existe_est = 0 THEN p_resultado := 'ERROR: Estudiante no existe'; RETURN; END IF;

    SELECT COUNT(*) INTO v_existe_cur FROM Curso WHERE id_curso = p_id_curso;
    IF v_existe_cur = 0 THEN p_resultado := 'ERROR: Curso no existe'; RETURN; END IF;

    SELECT COUNT(*) INTO v_ya_matriculado FROM Matricula
    WHERE id_estudiante = p_id_estudiante AND id_curso = p_id_curso;

    IF v_ya_matriculado > 0 THEN
        p_resultado := 'ERROR: Estudiante ya está matriculado en este curso'; RETURN;
    END IF;

    INSERT INTO Matricula VALUES (seq_matricula.NEXTVAL, SYSDATE, p_id_estudiante, p_id_curso);
    COMMIT;
    p_resultado := 'ÉXITO: Estudiante matriculado correctamente';
EXCEPTION WHEN OTHERS THEN p_resultado := 'ERROR: ' || SQLERRM;
END;
/
```

SISTEMA DE AUDITORÍA

```
CREATE TABLE Auditoria_Estudiante (
    id_auditoria NUMBER PRIMARY KEY,
    id_estudiante NUMBER,
    accion VARCHAR2(50),
    fecha TIMESTAMP,
    usuario VARCHAR2(100),
    datos_anteriores VARCHAR2(500)
);

CREATE SEQUENCE seq_auditoria START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER tr_auditoria_estudiantes
    AFTER UPDATE OR DELETE ON Estudiante FOR EACH ROW
DECLARE
    v_accion VARCHAR2(50);
    v_datos VARCHAR2(500);
BEGIN
```

```
139)     IF UPDATING THEN
140)         v_accion := 'ACTUALIZACIÓN';
141)         v_datos := 'Nombre: ' || :OLD.nombre || ' ' || :OLD.apellido
   || ', DNI: ' || :OLD.dni || ', Email: ' || :OLD.email;
142)     ELSIF DELETING THEN
143)         v_accion := 'ELIMINACIÓN';
144)         v_datos := 'Nombre: ' || :OLD.nombre || ' ' || :OLD.apellido
   || ', DNI: ' || :OLD.dni;
145)     END IF;
146)
147)     INSERT INTO Auditoria_Estudiante VALUES (
148)         seq_auditoria.NEXTVAL, :OLD.id_estudiante, v_accion, SYSTIMES
   TAMP, USER, v_datos
149)     );
150) END;
151) /
152)
153) CREATE OR REPLACE TRIGGER tr_validar_matricula
154)     BEFORE INSERT ON Matricula FOR EACH ROW
155) DECLARE
156)     v_curso_count NUMBER;
157) BEGIN
158)     SELECT COUNT(*) INTO v_curso_count FROM Curso WHERE id_curso = :N
   EW.id_curso;
159)     IF v_curso_count = 0 THEN
160)         RAISE_APPLICATION_ERROR(-20001, 'El curso especificado no exi
   ste');
161)     END IF;
162)     IF :NEW.fecha IS NULL THEN :NEW.fecha := SYSDATE; END IF;
163) END;
164) /
165)
```

VISTAS AVANZADAS

```
166) CREATE OR REPLACE VIEW vista_resumen_estudiantes AS
167) SELECT e.id_estudiante, e.nombre || ' ' || e.apellido as nombre_compl
   eto,
168)        e.dni, e.email, COUNT(m.id_matricula) as total_cursos
169) FROM Estudiante e LEFT JOIN Matricula m ON e.id_estudiante = m.id_est
   udiante
170) GROUP BY e.id_estudiante, e.nombre, e.apellido, e.dni, e.email;
171)
172) CREATE OR REPLACE VIEW vista_cursos_docentes AS
```

```
173) SELECT c.id_curso, c.nombre as nombre_curso, c.duracion_semanas, c.mo
   dalidad,
174)        d.nombre as nombre_docente, d.especialidad,
175)        (SELECT COUNT(*) FROM Matricula m WHERE m.id_curso = c.id_curs
   o) as estudiantes_inscritos
176) FROM Curso c JOIN Docente d ON c.id_docente = d.id_docente;
177)
178) CREATE OR REPLACE VIEW vista_estadisticas_sistema AS
179) SELECT (SELECT COUNT(*) FROM Estudiante) as total_estudiantes,
180)        (SELECT COUNT(*) FROM Docente) as total_docentes,
181)        (SELECT COUNT(*) FROM Curso) as total_cursos,
182)        (SELECT COUNT(*) FROM Matricula) as total_matriculas,
183)        (SELECT AVG(duracion_semanas) FROM Curso) as duracion_promedio
   _cursos,
184)        (SELECT COUNT(*) FROM Matricula m, Curso c WHERE m.id_curso =
   c.id_curso AND c.modalidad = 'Presencial') as matriculas_presencial,
185)        (SELECT COUNT(*) FROM Matricula m, Curso c WHERE m.id_curso =
   c.id_curso AND c.modalidad = 'Online') as matriculas_online
186) FROM dual;
187)
188) COMMIT;
```

2. ACTUALIZACION DE CODIGO JAVA

```
import java.sql.*;
import java.util.Scanner;

public class conexionBDoracle {

    private static final String URL =
"jdbc:oracle:thin:@localhost:1521:XE";
    private static final String USER = "Tecsup";
    private static final String PASSWORD = "Tecsup";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
            System.out.println("Conectado a la base de datos");

            int opcion;
            do {
                System.out.println("\n=== MENU PRINCIPAL ===");
```

```
                System.out.println("1. Insertar Estudiante");
                System.out.println("2. Actualizar Estudiante");
                System.out.println("3. Eliminar Estudiante");
                System.out.println("4. Ver Estudiantes");
                System.out.println("5. Ver Cursos");
                System.out.println("6. Consultas Avanzadas");
                System.out.println("7. PROCEDIMIENTOS ALMACENADOS");
                System.out.println("8. VISTAS AVANZADAS");
                System.out.println("9. AUDITORÍA");
                System.out.println("0. Salir");
                System.out.print("Opcion: ");

                opcion = scanner.nextInt();
                scanner.nextLine();

                switch (opcion) {
                    case 1 -> insertarEstudiante(conn, scanner);
                    case 2 -> actualizarEstudiante(conn, scanner);
                    case 3 -> eliminarEstudiante(conn, scanner);
                    case 4 -> verEstudiantes(conn);
                    case 5 -> verCursos(conn);
                    case 6 -> consultasAvanzadas(conn);
                    case 7 -> menuProcedimientosAlmacenados(conn,
scanner);
                    case 8 -> menuVistas(conn);
                    case 9 -> mostrarAuditoria(conn);
                    case 0 -> System.out.println("Saliendo...");
                    default -> System.out.println("Opcion no valida");
                }

            } while (opcion != 0);

        } catch (SQLException e) {
            System.out.println("Error de conexión: " +
e.getMessage());
        } finally {
            scanner.close();
        }
    }

 INSERTAR ESTUDIANTE
    public static void insertarEstudiante(Connection conn, Scanner
scanner) {
        try {
            System.out.print("Nombre: ");
            String nombre = scanner.nextLine();
            System.out.print("Apellido: ");
            String apellido = scanner.nextLine();
            System.out.print("DNI (8 dígitos): ");
            String dni = scanner.nextLine();
            System.out.print("Email: ");
            String email = scanner.nextLine();

            // Validar DNI antes de insertar
            if (dni.length() != 8 || !dni.matches("\\d+")) {
                System.out.println("ERROR: El DNI debe tener
exactamente 8 dígitos numéricos");
                return;
            }

            String sql = "INSERT INTO Estudiante VALUES
```

```
(seq_estudiante.NEXTVAL, ?, ?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, nombre);
            pstmt.setString(2, apellido);
            pstmt.setString(3, dni);
            pstmt.setString(4, email);

            pstmt.executeUpdate();
            System.out.println("Estudiante agregado correctamente");

        } catch (SQLException e) {
            System.out.println("Error al insertar: " +
e.getMessage());
        }
    }

    // ACTUALIZAR ESTUDIANTE
    public static void actualizarEstudiante(Connection conn, Scanner
scanner) {
        try {
            verEstudiantes(conn);
            System.out.print("ID del estudiante a actualizar: ");
            int id = scanner.nextInt();
            scanner.nextLine();

            System.out.print("Nuevo nombre: ");
            String nombre = scanner.nextLine();
            System.out.print("Nuevo email: ");
            String email = scanner.nextLine();

            String sql = "UPDATE Estudiante SET nombre = ?, email = ?
WHERE id_estudiante = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, nombre);
            pstmt.setString(2, email);
            pstmt.setInt(3, id);

            int filas = pstmt.executeUpdate();
            if (filas > 0) {
                System.out.println("Estudiante actualizado");
            } else {
                System.out.println("No se encontro el ID");
            }

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // ELIMINAR ESTUDIANTE
    public static void eliminarEstudiante(Connection conn, Scanner
scanner) {
        try {
            verEstudiantes(conn);
            System.out.print("ID del estudiante a eliminar: ");
            int id = scanner.nextInt();
            scanner.nextLine();

            // Primero eliminar matriculas
            String sql1 = "DELETE FROM Matricula WHERE id_estudiante =
?";
```

```
            PreparedStatement pstmt1 = conn.prepareStatement(sql1);
            pstmt1.setInt(1, id);
            pstmt1.executeUpdate();

            String sql2 = "DELETE FROM Estudiante WHERE id_estudiante
= ?";
            PreparedStatement pstmt2 = conn.prepareStatement(sql2);
            pstmt2.setInt(1, id);

            int filas = pstmt2.executeUpdate();
            if (filas > 0) {
                System.out.println("Estudiante eliminado");
            } else {
                System.out.println("No se encontro el ID");
            }

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // VER ESTUDIANTES
    public static void verEstudiantes(Connection conn) {
        try {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM
Estudiante");

            System.out.println("\n--- ESTUDIANTES ---");
            while (rs.next()) {
                System.out.println(rs.getInt("id_estudiante") + " | "
+
                        rs.getString("nombre") + " | " +
                        rs.getString("apellido") + " | " +
                        rs.getString("email"));
            }

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // VER CURSOS
    public static void verCursos(Connection conn) {
        try {
            String sql = "SELECT c.id_curso, c.nombre,
c.duracion_semanas, d.nombre as docente " +
                    "FROM Curso c JOIN Docente d ON c.id_docente =
d.id_docente";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            System.out.println("\n--- CURSOS ---");
            while (rs.next()) {
                System.out.println(rs.getInt("id_curso") + " | " +
                        rs.getString("nombre") + " | " +
                        rs.getInt("duracion_semanas") + " semanas | "
+
                        rs.getString("docente"));
            }
```

```
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // CONSULTAS AVANZADAS
    public static void consultasAvanzadas(Connection conn) {
        try {
            System.out.println("\n--- CONSULTA 1: Estudiantes con
cantidad de cursos ---");
            String sql1 = "SELECT e.nombre, e.apellido,
COUNT(m.id_matricula) as cursos " +
                    "FROM Estudiante e LEFT JOIN Matricula m ON
e.id_estudiante = m.id_estudiante " +
                    "GROUP BY e.id_estudiante, e.nombre, e.apellido";
            ejecutarConsulta(conn, sql1);

            System.out.println("\n--- CONSULTA 2: Todos los datos de
matriculas ---");
            String sql2 = "SELECT e.nombre, e.apellido, c.nombre as
curso, d.nombre as docente, m.fecha " +
                    "FROM Matricula m " +
                    "JOIN Estudiante e ON m.id_estudiante =
e.id_estudiante " +
                    "JOIN Curso c ON m.id_curso = c.id_curso " +
                    "JOIN Docente d ON c.id_docente = d.id_docente";
            ejecutarConsulta(conn, sql2);

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // MÉTODO AUXILIAR PARA EJECUTAR CONSULTAS
    public static void ejecutarConsulta(Connection conn, String sql)
throws SQLException {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);

        ResultSetMetaData meta = rs.getMetaData();
        int columnCount = meta.getColumnCount();

        // Mostrar columnas
        for (int i = 1; i <= columnCount; i++) {
            System.out.print(meta.getColumnName(i) + " | ");
        }
        System.out.println();

        while (rs.next()) {
            for (int i = 1; i <= columnCount; i++) {
                System.out.print(rs.getString(i) + " | ");
            }
            System.out.println();
        }

        rs.close();
        stmt.close();
    }

    // ========================================
    // ENTREGA 3 - NUEVOS MÉTODOS
```

```
    // ========================================

    // MENÚ PARA PROCEDIMIENTOS ALMACENADOS
    public static void menuProcedimientosAlmacenados(Connection conn,
Scanner scanner) {
        int opcion;
        do {
            System.out.println("\n=== PROCEDIMIENTOS ALMACENADOS
===");
            System.out.println("1. Estudiantes por Curso");
            System.out.println("2. Contar Estudiantes en Curso");
            System.out.println("3. Matricular Estudiante");
            System.out.println("0. Volver");
            System.out.print("Opcion: ");

            opcion = scanner.nextInt();
            scanner.nextLine();

            switch (opcion) {
                case 1 -> llamarProcedimientoEstudiantesCurso(conn,
scanner);
                case 2 -> llamarProcedimientoContarEstudiantes(conn,
scanner);
                case 3 ->
llamarProcedimientoMatricularEstudiante(conn, scanner);
                case 0 -> System.out.println("Volviendo al menú
principal...");
                default -> System.out.println("Opcion no valida");
            }

        } while (opcion != 0);
    }

    // PROCEDIMIENTO: ESTUDIANTES POR CURSO
    public static void llamarProcedimientoEstudiantesCurso(Connection
conn, Scanner scanner) {
        try {
            System.out.print("ID del curso: ");
            int idCurso = scanner.nextInt();

            // Versión simplificada usando consulta directa
            String sql = "SELECT e.id_estudiante, e.nombre,
e.apellido, e.email, e.dni " +
                    "FROM Estudiante e " +
                    "JOIN Matricula m ON e.id_estudiante =
m.id_estudiante " +
                    "WHERE m.id_curso = ?";

            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, idCurso);
            ResultSet rs = stmt.executeQuery();

            System.out.println("\n--- ESTUDIANTES EN EL CURSO " +
idCurso + " ---");
            boolean hayResultados = false;
            while (rs.next()) {
                hayResultados = true;
                System.out.println("ID: " + rs.getInt("id_estudiante")
+
                        " | Nombre: " + rs.getString("nombre") +
                        " " + rs.getString("apellido") +
```

```
                        " | Email: " + rs.getString("email") +
                        " | DNI: " + rs.getString("dni"));
            }

            if (!hayResultados) {
                System.out.println("No hay estudiantes matriculados en
este curso.");
            }

            rs.close();
            stmt.close();

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // PROCEDIMIENTO: CONTAR ESTUDIANTES EN CURSO
    public static void llamarProcedimientoContarEstudiantes(Connection
conn, Scanner scanner) {
        try {
            System.out.print("ID del curso: ");
            int idCurso = scanner.nextInt();

            // Versión simplificada usando consulta directa
            String sql = "SELECT COUNT(*) as total FROM Matricula
WHERE id_curso = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, idCurso);
            ResultSet rs = stmt.executeQuery();

            if (rs.next()) {
                int total = rs.getInt("total");
                System.out.println("Total de estudiantes en el curso "
+ idCurso + ": " + total);
            }

            rs.close();
            stmt.close();

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // PROCEDIMIENTO: MATRICULAR ESTUDIANTE
    public static void
llamarProcedimientoMatricularEstudiante(Connection conn, Scanner
scanner) {
        try {
            System.out.print("ID del estudiante: ");
            int idEstudiante = scanner.nextInt();
            System.out.print("ID del curso: ");
            int idCurso = scanner.nextInt();

            // Verificar si existe el estudiante
            String sqlCheckEst = "SELECT COUNT(*) FROM Estudiante
WHERE id_estudiante = ?";
            PreparedStatement stmtCheckEst =
conn.prepareStatement(sqlCheckEst);
            stmtCheckEst.setInt(1, idEstudiante);
```

```
            ResultSet rsEst = stmtCheckEst.executeQuery();
            rsEst.next();
            int existeEst = rsEst.getInt(1);
            rsEst.close();
            stmtCheckEst.close();

            if (existeEst == 0) {
                System.out.println("ERROR: El estudiante no existe");
                return;
            }

            // Verificar si existe el curso
            String sqlCheckCur = "SELECT COUNT(*) FROM Curso WHERE
id_curso = ?";
            PreparedStatement stmtCheckCur =
conn.prepareStatement(sqlCheckCur);
            stmtCheckCur.setInt(1, idCurso);
            ResultSet rsCur = stmtCheckCur.executeQuery();
            rsCur.next();
            int existeCur = rsCur.getInt(1);
            rsCur.close();
            stmtCheckCur.close();

            if (existeCur == 0) {
                System.out.println("ERROR: El curso no existe");
                return;
            }

            // Verificar si ya está matriculado
            String sqlCheckMat = "SELECT COUNT(*) FROM Matricula WHERE
id_estudiante = ? AND id_curso = ?";
            PreparedStatement stmtCheckMat =
conn.prepareStatement(sqlCheckMat);
            stmtCheckMat.setInt(1, idEstudiante);
            stmtCheckMat.setInt(2, idCurso);
            ResultSet rsMat = stmtCheckMat.executeQuery();
            rsMat.next();
            int yaMatriculado = rsMat.getInt(1);
            rsMat.close();
            stmtCheckMat.close();

            if (yaMatriculado > 0) {
                System.out.println("ERROR: El estudiante ya está
matriculado en este curso");
                return;
            }

            // Realizar matrícula
            String sqlInsert = "INSERT INTO Matricula (id_matricula,
id_estudiante, id_curso, fecha) " +
                    "VALUES (seq_matricula.NEXTVAL, ?, ?, SYSDATE)";
            PreparedStatement stmtInsert =
conn.prepareStatement(sqlInsert);
            stmtInsert.setInt(1, idEstudiante);
            stmtInsert.setInt(2, idCurso);
            stmtInsert.executeUpdate();
            stmtInsert.close();

            System.out.println("ÉXITO: Estudiante matriculado
correctamente");
```

```
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // MENÚ PARA VISTAS AVANZADAS
    public static void menuVistas(Connection conn) {
        try {
            System.out.println("\n=== VISTAS AVANZADAS ===");

            System.out.println("\n--- RESUMEN DE ESTUDIANTES ---");
            String sql1 = "SELECT * FROM vista_resumen_estudiantes";
            ejecutarConsulta(conn, sql1);

            System.out.println("\n--- CURSOS CON DOCENTES ---");
            String sql2 = "SELECT * FROM vista_cursos_docentes";
            ejecutarConsulta(conn, sql2);

            System.out.println("\n--- ESTADÍSTICAS DEL SISTEMA ---");
            String sql3 = "SELECT * FROM vista_estadisticas_sistema";
            ejecutarConsulta(conn, sql3);

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    // MOSTRAR AUDITORÍA
    public static void mostrarAuditoria(Connection conn) {
        try {
            String sql = "SELECT * FROM Auditoria_Estudiante ORDER BY
fecha DESC";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            System.out.println("\n=== AUDITORÍA DE ESTUDIANTES ===");
            boolean hayRegistros = false;
            while (rs.next()) {
                hayRegistros = true;
                System.out.println("ID: " + rs.getInt("id_auditoria")
+
                        " | Estudiante: " + rs.getInt("id_estudiante")
+
                        " | Acción: " + rs.getString("accion") +
                        " | Fecha: " + rs.getTimestamp("fecha") +
                        " | Usuario: " + rs.getString("usuario") +
                        " | Datos: " +
rs.getString("datos_anteriores"));
            }

            if (!hayRegistros) {
                System.out.println("No hay registros de auditoría
aún.");
            }

            rs.close();
            stmt.close();

        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
```

```
        }
}
```

CONCLUSIONES

a) Base de datos funcional y Todo el SQL funciona correctamente

b) Java conectado al 100% y La aplicación se comunica sin errores

c) Procedimientos almacenados implementados con Consultas optimizadas

d) Triggers activos y un sistema de auditoría automática

e) Vistas operativas con Reportes y consultas avanzadas

f) CRUD completo - Todas las operaciones básicas funcionando

Herramientas utilizadas para el desarrollo del proyecto

a) Oracle Database 11g y PL/SQL

b) Java y JDBC

c) Características avanzadas del proyecto: Procedimientos, Triggers, Vistas, Auditoría

ENLACE DE TODO EL PROYECTO EN EL GitHub

greciahuanco/1er_avance_BD