```
package test;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Calendar;
import java.util.Collection;
import java.util.GregorianCalendar;

import dao.AcademiaDAO;
import dao.AcademiaDAOImplJDBC;
import entidades.Alumno;
import entidades.Curso;
import entidades.Matricula;

public class InsertarHelper {

    private AcademiaDAO dao=null;

    // Constructor
    public InsertarHelper() {
        System.out.println("Creando el DAO...");
        dao=new AcademiaDAOImplJDBC();
    }

    /*
     * Alumnos
     */
    private void insertarAlumno(int id, String nombre, String rutaFoto) {

        System.out.println("\nCreando un alumno...");
        Alumno alumno=new Alumno(id,nombre);

        // Leemos la foto del disco, la guardamos en el
        // objeto Alumno y posteriormente se graba en la BD
        File file = new File(rutaFoto);

        byte[] foto=null;
        try {
            foto = getBytesFromFile(file);
        } catch (IOException e) { e.printStackTrace(); }

        alumno.setFoto(foto);

        System.out.println("Grabando el nuevo alumno...");
        if (dao.grabarAlumno(alumno)==1) {
            System.out.print("Se ha grabado el alumno");
        } else {
            System.out.print("Error al grabar el alumno");
        }
    }

    private void modificarAlumno(int id, String nombre, String rutaFoto) {

        // Recuperamos al alumno a partir de su id
        Alumno alumno=dao.getAlumno(id);

        System.out.println("\nModificando el nombre del alumno con id: "+id+" y
```

```
nombre: "+alumno.getNombreAlumno());
        alumno.setNombreAlumno(nombre);

        // Si se ha pasado la ruta de la foto...
        if (rutaFoto!=null) {
            System.out.println("\nModificando la foto del alumno con id: "+id+" y
nombre: "+alumno.getNombreAlumno());
            // Leemos la foto del disco, la guardamos en el
            // objeto Alumno y posteriormente se graba en la BD
            File file = new File(rutaFoto);

            byte[] foto=null;
            try {
                foto = getBytesFromFile(file);
            } catch (IOException e) { e.printStackTrace(); }
            alumno.setFoto(foto);
        }

        if (dao.actualizarAlumno(alumno)==1) {
            System.out.print("Se ha modificado el alumno con id: "+id);
        } else {
            System.out.print("Error al modificar el alumno con id: "+id);
        }
    }

    /*
     * Cursos
     */

    private void insertarCurso(int id, String nombre) {

        System.out.println("\nCreando un curso...");
        Curso curso=new Curso(id,nombre);

        System.out.println("Grabando el nuevo curso...");
        if (dao.grabarCurso(curso)==1) {
            System.out.print("Se ha grabado el curso");
        } else {
            System.out.print("Error al grabar el curso");
        }
    }


    private void modificarCurso(int id, String nombre) {

        // Recuperamos al curso a partir de su id
        Curso curso=dao.getCurso(id);

        System.out.println("\nModificando el nombre del curso con id: "+id+" y
nombre: "+curso.getNombreCurso());
        curso.setNombreCurso(nombre);

        if (dao.actualizarCurso(curso)==1) {
            System.out.print("Se ha modificado el curso con id: "+id);
        } else {
            System.out.print("Error al modificar el curso con id: "+id);
        }
    }
```

```
    /*
     * Matriculas
     */

    private void insertarMatricula(int id_alumno, int id_curso) {

        System.out.println("\nCreando una matricula...");
        Matricula matricula=new Matricula(new java.util.Date());

        // Recuperamos al alumno a partir de su id
        Alumno alumno=dao.getAlumno(id_alumno);

        // Recuperamos al curso a partir de su id
        Curso curso=dao.getCurso(id_curso);

        matricula.setAlumno(alumno);
        matricula.setCurso(curso);

        System.out.println("Grabando la nueva matricula...");
        if (dao.grabarMatricula(matricula)==1) {
            System.out.print("Se ha grabado la matricula");
        } else {
            System.out.print("Error al grabar la matricula");
        }
    }

    private void modificarMatricula(int id_alumno, int id_curso, java.util.Date
fecha) {

        // A partir del id del alumno y del id del curso
        // obtenemos el id de matricula
        long id_matricula=dao.getIdMatricula(id_alumno, id_curso);

        // Mediante el id de matricula obtenemos la matricula
        // que queremos modificar
        Matricula matricula=dao.getMatricula(id_matricula);

        // Cambiamos la fecha de inicio de la matricula
        matricula.setFechaInicio(fecha);

        System.out.println("\nModificando fecha de la matricula...");
        if (dao.actualizarMatricula(matricula)==1) {
            System.out.print("Se ha modificado la matricula");
        } else {
            System.out.print("Error al modificar la matricula");
        }
    }

    private void showAllData() {
        showData(dao.cargarAlumnos(),"Alumnos");
        showData(dao.cargarCursos(),"Cursos");
        showData(dao.cargarMatriculas(),"Matriculas");
    }

    private void showData(Collection<?> coleccion, String entidad) {

        System.out.println("\nMostrando..."+entidad);

        for (Object obj:coleccion)
```

```
        System.out.println(obj);
}

public static void main(String[] args) {

    InsertarHelper programa=new InsertarHelper();

    /*
     * Insertar alumnos
     */

    programa.insertarAlumno(1000,"Daniel","imgs/cara2.jpg");
    programa.insertarAlumno(1001,"Francisco","imgs/cara4.jpg");

    // Cambiarle el nombre al primer alumno creado
    programa.modificarAlumno(1000, "Ezequiel",null);

    // Volverle a cambiar el nombre y ahora la foto
    programa.modificarAlumno(1000, "Agapito","imgs/cara1.jpg");

    /*
     * Insertar cursos
     */
    programa.insertarCurso(500, "Java");
    programa.insertarCurso(501, ".NET");

    // Modificar el curso creado
    programa.modificarCurso(500, "Java avanzado");


    /*
     * Insertar matriculas
     */
    programa.insertarMatricula(1000, 500);
    programa.insertarMatricula(1000, 501);
    programa.insertarMatricula(1001, 500);

    /*
     * Modificar fecha de la segunda matricula
     */
    Calendar fecha=GregorianCalendar.getInstance();
    fecha.set(Calendar.MONTH, 11);
    programa.modificarMatricula(1001, 500, fecha.getTime());

    /*
     * Mostrar lo que hemos grabado
     *
     */
    programa.showAllData();

    System.out.println("\nfin del programa.");
}

/*
 * Devuelve el contenido del fichero (la foto)
 * en un array de bytes
 */

private static byte[] getBytesFromFile(File file) throws IOException {
```

```
        InputStream is = new FileInputStream(file);

        // Obtener el tamaÃ±o del fichero
        long length = file.length();

        // No podemos crear un array usando un tipo long.
        // Es necesario que sea un tipo int.
        // Antes de convertirlo a int, comprobamos
        // que el fichero no es mayor que Integer.MAX_VALUE
        if (length > Integer.MAX_VALUE) {
            System.out.println("Fichero demasiado grande!");
            System.exit(1);
        }

        // Creamos el byte array que almacenarÃ¡
        // temporalmente los datos leidos
        byte[] bytes = new byte[(int)length];

        // Leemos
        int offset = 0;
        int numRead = 0;
        while (offset < bytes.length
                && (numRead=is.read(bytes, offset, bytes.length-offset)) >= 0) {
            offset += numRead;
        }

        // Comprobacion de que todos los bytes se han leido
        if (offset < bytes.length) {
            throw new IOException("No se ha podido leer complemtamente el fichero
"+file.getName());
        }

        // Cerrar el input stream y devolver los bytes
        is.close();
        return bytes;
    }

}
package test;

import dao.AcademiaDAO;
import dao.AcademiaDAOImplJDBC;

public class BorrarHelper {

    private AcademiaDAO dao;

    // Constructor
    public BorrarHelper() {
        System.out.println("Creando el DAO...");
        dao = new AcademiaDAOImplJDBC();
    }

    /*
     * Eliminar Alumno
     */
    private void borrarAlumno(int id) {
        System.out.println("\nEliminando el alumno con ID: " + id);
        if (dao.borrarAlumno(id) == 1) {
```

```
            System.out.println("Se ha eliminado el alumno con ID: " + id);
        } else {
            System.out.println("Error al eliminar el alumno con ID: " + id);
        }
    }

    /*
     * Eliminar Curso
     */
    private void borrarCurso(int id) {
        System.out.println("\nEliminando el curso con ID: " + id);
        if (dao.borrarCurso(id) == 1) {
            System.out.println("Se ha eliminado el curso con ID: " + id);
        } else {
            System.out.println("Error al eliminar el curso con ID: " + id);
        }
    }

    /*
     * Eliminar MatrÃcula
     */
    private void borrarMatricula(int id_alumno, int id_curso) {
        System.out.println("\nEliminando la matrÃcula del alumno " + id_alumno + "
en el curso " + id_curso);
        long id_matricula = dao.getIdMatricula(id_alumno, id_curso);
        if (dao.borrarMatricula(id_matricula) == 1) {
            System.out.println("Se ha eliminado la matrÃcula correctamente.");
        } else {
            System.out.println("Error al eliminar la matrÃcula.");
        }
    }

    public static void main(String[] args) {
        BorrarHelper programa = new BorrarHelper();


        // Eliminar matrÃculas
        programa.borrarMatricula(1000, 500);
        programa.borrarMatricula(1000, 501);
        programa.borrarMatricula(1001, 500);

        // Eliminar alumnos
        programa.borrarAlumno(1000);
        programa.borrarAlumno(1001);

        // Eliminar cursos
        programa.borrarCurso(500);
        programa.borrarCurso(501);


        System.out.println("\nProceso de eliminaciÃ³n finalizado.");
    }
}
package entidades;

import java.io.Serializable;
import java.util.Date;

public class Matricula implements Serializable {
```

```
    private static final long serialVersionUID = 1L;

    private int id;
    private Alumno alumno;
    private Curso curso;
    private Date fechaInicio;

    public Matricula() {}

    public Matricula(Date fechaInicio) {
        this.fechaInicio = fechaInicio;
    }

    public Matricula(int idMatricula, Alumno alumno, Curso curso, java.sql.Date
fechaInicio) {
        id = idMatricula;
        this.alumno = alumno;
        this.curso = curso;
        this.fechaInicio = fechaInicio;
    }

    public int getId() {
        return id;
    }

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

    public Curso getCurso() {
        return curso;
    }

    public void setCurso(Curso curso) {
        this.curso = curso;
    }

    public Alumno getAlumno() {
        return alumno;
    }

    public void setAlumno(Alumno alumno) {
        this.alumno = alumno;
    }

    public Date getFechaInicio() {
        return fechaInicio;
    }

    public void setFechaInicio(Date fechaInicio) {
        this.fechaInicio = fechaInicio;
    }

    @Override
    public String toString() {
        return "Matricula{" +
                "alumno=" + alumno +
                ", curso=" + curso +
```

```
                ", fechaInicio=" + fechaInicio +
                '}';
    }
}
package entidades;

import java.io.Serializable;

public class Curso implements Serializable {

    private static final long serialVersionUID = 1L;

    private int idCurso;
    private String nombreCurso;

    public Curso() {}

    public Curso(int idCurso, String nombreCurso) {
        this.idCurso = idCurso;
        this.nombreCurso = nombreCurso;
    }

    public int getIdCurso() {
        return idCurso;
    }

    public void setIdCurso(int idCurso) {
        this.idCurso = idCurso;
    }

    public String getNombreCurso() {
        return nombreCurso;
    }

    public void setNombreCurso(String nombreCurso) {
        this.nombreCurso = nombreCurso;
    }

    @Override
    public String toString() {
        return idCurso + " - " + nombreCurso;
    }
}
package entidades;

import java.io.Serializable;

public class Alumno implements Serializable {

    private static final long serialVersionUID = 1L;

    private int idAlumno;
    private String nombreAlumno;
    private byte[] foto;

    public byte[] getFoto() {
        return foto;
    }
```

```
    public void setFoto(byte[] foto) {
        this.foto = foto;
    }

    public Alumno() {}

    public Alumno(int idAlumno, String nombreAlumno) {
        this.idAlumno = idAlumno;
        this.nombreAlumno = nombreAlumno;
    }

    public int getIdAlumno() {
        return this.idAlumno;
    }

    public void setIdAlumno(int idAlumno) {
        this.idAlumno = idAlumno;
    }

    public String getNombreAlumno() {
        return this.nombreAlumno;
    }

    public void setNombreAlumno(String nombreAlumno) {
        this.nombreAlumno = nombreAlumno;
    }

    public String toString() {
        return this.idAlumno+" - "+this.nombreAlumno;
    }

}
package dao;

import java.io.ByteArrayInputStream;
import java.sql.*;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import entidades.Alumno;
import entidades.Curso;
import entidades.Matricula;

public class AcademiaDAOImplJDBC implements AcademiaDAO {
    // Cadena de conexiÃ³n predeterminada
    private String URLConexion = "jdbc:mysql://localhost:3306/dbformacion?
user=dam2&password=dam2";

    /*
     * SQL QUERIES
     */
    private static final String FIND_ALL_ALUMNOS_SQL = "SELECT id_alumno,
nombre_alumno, foto FROM alumnos";
    private static final String FIND_ALUMNO_BY_ID_SQL = "SELECT id_alumno,
nombre_alumno, foto FROM alumnos WHERE id_alumno = ?";
    private static final String INSERT_ALUMNO_SQL = "INSERT INTO alumnos
(id_alumno, nombre_alumno, foto) VALUES (?, ?, ?)";
    private static final String UPDATE_ALUMNO_SQL = "UPDATE alumnos SET
```

```
nombre_alumno = ?, foto = ? WHERE id_alumno = ?";
    private static final String DELETE_ALUMNO_SQL = "DELETE FROM alumnos WHERE
id_alumno = ?";

    private static final String FIND_ALL_CURSOS_SQL = "SELECT id_curso,
nombre_curso FROM cursos";
    private static final String FIND_CURSO_BY_ID_SQL = "SELECT id_curso,
nombre_curso FROM cursos WHERE id_curso = ?";
    private static final String INSERT_CURSO_SQL = "INSERT INTO cursos (id_curso,
nombre_curso) VALUES (?, ?)";
    private static final String UPDATE_CURSO_SQL = "UPDATE cursos SET nombre_curso
= ? WHERE id_curso = ?";
    private static final String DELETE_CURSO_SQL = "DELETE FROM cursos WHERE
id_curso = ?";

    private static final String FIND_ALL_MATRICULAS_SQL = "SELECT id_matricula,
id_alumno, id_curso, fecha_inicio FROM matriculas";
    private static final String FIND_MATRICULA_BY_ALUMNO_CURSO = "SELECT
id_matricula, id_alumno, id_curso, fecha_inicio FROM matriculas WHERE id_alumno = ?
AND id_curso = ?";
    private static final String FIND_MATRICULA_BY_ID_SQL = "SELECT id_matricula,
id_alumno, id_curso, fecha_inicio FROM matriculas WHERE id_matricula = ?";
    private static final String INSERT_MATRICULA_SQL = "INSERT INTO matriculas
(id_alumno, id_curso, fecha_inicio) VALUES (?, ?, ?)";
    private static final String UPDATE_MATRICULA_SQL = "UPDATE matriculas SET
id_alumno = ?, id_curso = ?, fecha_inicio = ? WHERE id_matricula = ?";
    private static final String DELETE_MATRICULA_SQL = "DELETE FROM matriculas
WHERE id_matricula = ?";

    /*
     * CONSTRUCTORES
     */
    public AcademiaDAOImplJDBC() {}

    public AcademiaDAOImplJDBC(String URLConexion) {
        this.URLConexion = URLConexion;
    }

    /*
     * OPERACIONES GENERALES
     */
    private Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URLConexion);
    }

    private void releaseConnection(Connection con) {
        if (con != null) {
            try {
                con.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    /*
     * OPERACIONES ALUMNO
     */
    @Override
```

```
    public Collection<Alumno> cargarAlumnos() {
        Collection<Alumno> alumnos = new ArrayList<>();
        Connection con = null;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(FIND_ALL_ALUMNOS_SQL);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                int id = rs.getInt("id_alumno");
                String nombre = (rs.getString("nombre_alumno") != null ?
rs.getString("nombre_alumno") : "Sin nombre");
                Blob foto = rs.getBlob("foto");
                Alumno al = new Alumno(id, nombre);
                alumnos.add(al);
                if (foto != null) al.setFoto(foto.getBytes(1L, (int)
foto.length()));
                else al.setFoto(null);
            }
            rs.close();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return alumnos;
    }

    @Override
    public Alumno getAlumno(int idAlumno) {
        Connection con = null;
        Alumno alumno = null;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(FIND_ALUMNO_BY_ID_SQL);
            ps.setInt(1, idAlumno);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                int id = rs.getInt("id_alumno");
                String nombre = rs.getString("nombre_alumno");
                Blob foto = rs.getBlob("foto");
                alumno = new Alumno(id, nombre);
                if (foto != null) alumno.setFoto(foto.getBytes(1L, (int)
foto.length()));
                else alumno.setFoto(null);
            }
            rs.close();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return alumno;
    }

    @Override
    public int grabarAlumno(Alumno alumno) {
        Connection con = null;
```

```
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(INSERT_ALUMNO_SQL);
            ps.setInt(1, alumno.getIdAlumno());
            ps.setString(2, alumno.getNombreAlumno());

            if (alumno.getFoto()!=null) ps.setBinaryStream(3, new
ByteArrayInputStream(alumno.getFoto()));
            else ps.setBinaryStream(3, null);

            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public int actualizarAlumno(Alumno alumno) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(UPDATE_ALUMNO_SQL);
            ps.setString(1, alumno.getNombreAlumno());
            // foto en binario o no
            if (alumno.getFoto() != null) ps.setBinaryStream(2, new
ByteArrayInputStream(alumno.getFoto()));
            else ps.setBinaryStream(2, null);
            // Id alumno
            ps.setInt(3, alumno.getIdAlumno());


            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public int borrarAlumno(int idAlumno) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(DELETE_ALUMNO_SQL);
            ps.setInt(1, idAlumno);
            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
```

```
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public Collection<Curso> cargarCursos() {
        Collection<Curso> cursos = new ArrayList<>();
        Connection con = null;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(FIND_ALL_CURSOS_SQL);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                int id = rs.getInt("id_curso");
                String nombre = rs.getString("nombre_curso");
                cursos.add(new Curso(id, nombre));
            }
            rs.close();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return cursos;
    }

    @Override
    public Curso getCurso(int idCurso) {
        Connection con = null;
        Curso curso = null;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(FIND_CURSO_BY_ID_SQL);
            ps.setInt(1, idCurso);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                String nombre = rs.getString("nombre_curso");
                curso = new Curso(idCurso, nombre);
            }
            rs.close();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return curso;
    }

    @Override
    public int grabarCurso(Curso curso) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(INSERT_CURSO_SQL);
```

```
            ps.setInt(1, curso.getIdCurso());
            ps.setString(2, curso.getNombreCurso());
            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public int actualizarCurso(Curso curso) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(UPDATE_CURSO_SQL);
            ps.setString(1, curso.getNombreCurso());
            ps.setInt(2, curso.getIdCurso());
            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public int borrarCurso(int idCurso) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(DELETE_CURSO_SQL);
            ps.setInt(1, idCurso);
            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public Collection<Matricula> cargarMatriculas() {
        Collection<Matricula> matriculas = new ArrayList<>();
        Connection con = null;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(FIND_ALL_MATRICULAS_SQL);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Matricula matricula = getMatricula(rs.getInt("id_matricula"));
```

```
//                Alumno alumno = getAlumno(rs.getInt("id_alumno"));
//                Curso curso = getCurso(rs.getInt("id_curso"));

//                Date fechaInicio = rs.getDate("fecha_inicio");
                matriculas.add(matricula);
            }
            rs.close();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return matriculas;
    }

    @Override
    public long getIdMatricula(int idAlumno, int idCurso) {
        Connection con = null;
        long idMatricula = -1;
        try {
            con = getConnection();
            PreparedStatement ps =
con.prepareStatement(FIND_MATRICULA_BY_ALUMNO_CURSO);
            ps.setInt(1, idAlumno);
            ps.setInt(2, idCurso);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                idMatricula = rs.getLong("id_matricula");
            }
            rs.close();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return idMatricula;
    }

    @Override
    public Matricula getMatricula(long idMatricula) {
        Connection con = null;
        Matricula matricula = null;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(FIND_MATRICULA_BY_ID_SQL);
            // "SELECT id_matricula, id_alumno, id_curso, fecha_inicio FROM
matriculas WHERE id_matricula = ?
            ps.setLong(1, idMatricula);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                Alumno alumno = getAlumno(rs.getInt("id_alumno"));
                Curso curso = getCurso(rs.getInt("id_curso"));
                Date fechaInicio = rs.getDate("fecha_inicio");
                matricula = new Matricula((int) idMatricula, alumno, curso,
fechaInicio);
            }
            rs.close();
```

```
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return matricula;
    }

    @Override
    public int grabarMatricula(Matricula matricula) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(INSERT_MATRICULA_SQL);
            ps.setInt(1, matricula.getAlumno().getIdAlumno());
            ps.setInt(2, matricula.getCurso().getIdCurso());
            ps.setDate(3, new java.sql.Date(matricula.getFechaInicio().getTime()));
            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public int actualizarMatricula(Matricula matricula) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(UPDATE_MATRICULA_SQL);
            ps.setInt(1, matricula.getAlumno().getIdAlumno());
            ps.setInt(2, matricula.getCurso().getIdCurso());
            ps.setDate(3, new java.sql.Date(matricula.getFechaInicio().getTime()));
            ps.setInt(4, matricula.getId());
            filasAfectadas = ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }

    @Override
    public int borrarMatricula(long idMatricula) {
        Connection con = null;
        int filasAfectadas = 0;
        try {
            con = getConnection();
            PreparedStatement ps = con.prepareStatement(DELETE_MATRICULA_SQL);
            ps.setLong(1, idMatricula);
            filasAfectadas = ps.executeUpdate();
```

```
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            releaseConnection(con);
        }
        return filasAfectadas;
    }
}
```

```
package dao;

import java.util.Collection;
import entidades.Alumno;
import entidades.Curso;
import entidades.Matricula;

public interface AcademiaDAO {

    public Collection<Alumno> cargarAlumnos();
    public Alumno getAlumno(int idAlumno);
    public int grabarAlumno(Alumno alumno);
    public int actualizarAlumno(Alumno alumno);
    public int borrarAlumno(int idAlumno);

    public Collection<Curso> cargarCursos();
    public Curso getCurso(int idCurso);
    public int grabarCurso(Curso curso);
    public int actualizarCurso(Curso curso);
    public int borrarCurso(int idCurso);

    public Collection<Matricula> cargarMatriculas();
    public long getIdMatricula(int idAlumno, int idCurso);
    public Matricula getMatricula(long idMatricula);
    public int grabarMatricula(Matricula matricula);
    public int actualizarMatricula(Matricula matricula);
    public int borrarMatricula(long idMatricula);

}
```