

**Proyecto Innovación Creación**

**Tema:** Sistema de administración de un colegio

**­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Nombres y Apellidos del estudiante**

Luis Nicolás Moya Ocampo

Felipe Andrés Muñoz Cuenca

Nelson Ramiro Tipán Brito

**Docente Tutor**

**Nombres y Apellidos del docente tutor**

Oscar Acero

**Carrera:** Desarrollo de software

D.M Quito, 8 de septiembre del 2024

# Índice

[Índice 2](#_Toc176533778)

[Índice de Anexos 4](#_Toc176533779)

[Resumen 5](#_Toc176533780)

[Introducción 6](#_Toc176533781)

[Nombre del proyecto 6](#_Toc176533782)

[Marco contextual 6](#_Toc176533783)

[Formulación y Definición del Problema 6](#_Toc176533784)

[Idea a defender 6](#_Toc176533785)

[Objeto de estudio y campo de acción 7](#_Toc176533786)

[Objeto de estudio 7](#_Toc176533787)

[Campo de acción 7](#_Toc176533788)

[Justificación 8](#_Toc176533789)

[Objetivos 9](#_Toc176533790)

[General 9](#_Toc176533791)

[Específicos 9](#_Toc176533792)

[Tipo de investigación 10](#_Toc176533793)

[Métodos de investigación 10](#_Toc176533794)

[Técnicas e instrumentos de investigación 11](#_Toc176533795)

[Implementación 11](#_Toc176533796)

[Resultados esperados 11](#_Toc176533797)

[Desarrollo de la propuesta 11](#_Toc176533798)

[Fundamentación Teórica 12](#_Toc176533799)

[Elaboración de la propuesta 15](#_Toc176533800)

[Pseudocódigo 15](#_Toc176533801)

[Código 19](#_Toc176533802)

[Capturas de la correcta ejecución 19](#_Toc176533803)

[Conclusiones 19](#_Toc176533804)

[Recomendaciones 20](#_Toc176533805)

[Link del Repositorio con avances del programa 21](#_Toc176533806)

[Referencias 22](#_Toc176533807)

[Anexo 1 Esquema del problema 24](#_Toc176533808)

[Formulación y Definición del Problema 24](#_Toc176533809)

[Anexo 2 25](#_Toc176533810)

[Código del programa 25](#_Toc176533811)

[Anexo 3 82](#_Toc176533812)

[Capturas de la correcta ejecución 82](#_Toc176533813)

[Anexo 4 82](#_Toc176533814)

[Link del Repositorio 82](#_Toc176533815)

# Índice de Anexos

[Anexo 1 24](#_Toc176533816)

[Anexo 2 25](#_Toc176533817)

[Anexo 3 82](#_Toc176533818)

[Anexo 4 82](#_Toc176533819)

# Resumen

En el proyecto, se abordó la necesidad de optimizar la gestión escolar mediante el desarrollo de un sistema de administración de un colegio. El problema identificado radicaba en la falta de una herramienta eficiente para gestionar la información de los alumnos, sus cursos y materias. Se estableció como objetivo principal la creación de un programa en Java que permitiera insertar, cambiar de curso, eliminar y modificar datos de los alumnos de manera intuitiva. Durante la segunda fase del proyecto, se integró una base de datos MySQL para almacenar y gestionar de manera más eficiente los registros, permitiendo la persistencia y accesibilidad de la información. La hipótesis subyacente sostenía que la implementación de este sistema, ahora con soporte de base de datos, facilita la administración escolar y mejora la accesibilidad de la información. La metodología empleada consistió en el diseño y programación del software, incorporando funciones para la manipulación de datos de estudiantes y la gestión de materias, ahora con soporte para consultas SQL que facilitan la interacción con la base de datos. A través de consultas por ID, el programa posibilita acceder a la información detallada de cada alumno, tanto desde la interfaz del programa como directamente desde la base de datos. En resumen, la propuesta desarrollada ofrece una solución integral para la administración eficiente de un colegio, abordando desde la identificación del problema hasta la implementación de un sistema que cumple con los objetivos planteados.

***Palabras claves:*** gestión escolar, MySQL, sistema de administración.

# Introducción

## Nombre del proyecto

**Tema: “**Sistema de administración de un colegio”

## Marco contextual

Se identificó la necesidad de mejorar la gestión escolar como respuesta a una situación problemática. Los antecedentes revelan carencias en la administración de la información académica, afectando a alumnos, docentes y personal administrativo. Los involucrados comprenden la comunidad educativa en su conjunto. La ubicación geográfica, situada en un entorno urbano, destaca la importancia de una solución eficiente. Desde el punto de vista sociocultural y económico, el problema impacta el desarrollo humano al comprometer la calidad educativa. La deficiencia en la gestión repercute en el desenvolvimiento diario de las actividades del sector educativo, generando desafíos adicionales para la comunidad escolar. La implementación de un sistema de administración escolar apoyado por una base de datos permite un manejo más eficiente y seguro de la información.

## Formulación y Definición del Problema

La gráfica representa de manera visual las causas y consecuencias asociadas a la implementación del sistema de administración escolar en el contexto educativo. (**Anexo 1**)

## Idea a defender

La implementación eficiente de un sistema centralizado de administración escolar mediante un programa en Java, con funcionalidades específicas de inserción, modificación y eliminación de datos de alumnos, así como gestión de materias, permitirá superar la falta de estructura y desorganización en la gestión académica del colegio. La integración con una base de datos MySQL refuerza esta estructura al garantizar la persistencia de los datos, la seguridad en su manejo, y la capacidad de realizar consultas avanzadas. La idea principal es garantizar la accesibilidad y disponibilidad de información clave, facilitando una toma de decisiones informada, mejorando el desempeño académico y optimizando los procesos administrativos. La defensa de esta idea se basa en la creación y aplicación efectiva de un sistema que aborda las causas fundamentales de la ineficiencia académica, promoviendo así un entorno educativo más eficiente y de mayor calidad.

## Objeto de estudio y campo de acción

### Objeto de estudio

El problema a resolver se centra en la falta de una estructura eficiente que permita gestionar datos de manera organizada y accesible en un colegio. Hasta el momento, se reconoce que la gestión manual y descentralizada de información ha generado desafíos en la toma de decisiones, afectando el desempeño académico y la eficiencia operativa. La investigación busca comprender a fondo las necesidades y dinámicas específicas de la comunidad escolar, identificar las limitaciones del sistema actual y proponer soluciones efectivas a través de la implementación de un sistema de administración escolar que ahora incluye la integración de una base de datos para un manejo más robusto de la información.

### Campo de acción

La investigación se enfocó en implementar y evaluar un sistema de administración escolar en un periodo determinado, por medio de la programación en Java y la integración de una base de datos MySQL, adaptándose a las circunstancias y requerimientos particulares de la comunidad educativa. La delimitación espacial se delimita al colegio en cuestión, abordando las problemáticas y proponiendo soluciones contextualizadas en este entorno educativo específico.

## Justificación

La investigación sobre la implementación de un sistema de administración escolar en un colegio justifica su relevancia al impactar directamente en la mejora de la gestión académica y beneficiar a diversos actores en la comunidad educativa. Los estudiantes pueden experimentar un entorno más organizado, facilitando un seguimiento preciso de su desempeño académico. El personal educativo y administrativo se puede beneficiar al contar con herramientas tecnológicas que agilizan la gestión de datos y permiten una toma de decisiones más informada. La incorporación de una base de datos MySQL asegura que los datos no solo se manejen de manera eficiente sino también segura y persistente, lo que aporta una ventaja considerable en comparación con la gestión manual anterior. La investigación optimiza procesos, reduce tiempos operativos y mejora la accesibilidad a información clave, aportando significativamente al colegio.

El valor de la investigación radica en su enfoque específico en las necesidades de la institución, adaptando la solución a su contexto único. La pertinencia del estudio destaca al abordar las causas de la ineficiencia académica y proponer soluciones prácticas y tecnológicas. Científicamente, contribuye al campo de la administración escolar al desarrollar un sistema que puede servir como modelo para otras instituciones educativas. La relevancia tecnológica se evidencia en la implementación de un programa en Java con una base de datos MySQL, introduciendo soluciones innovadoras para mejorar la eficiencia y calidad en la gestión académica. Moderniza procesos internos y fomenta el uso de tecnologías educativas, preparando a los estudiantes para entornos tecnológicamente avanzados. En resumen, la investigación no solo resuelve problemas inmediatos, sino que establece un precedente para la integración efectiva de la tecnología en la gestión académica, beneficiando ampliamente a la educación.

## Objetivos

### General

Desarrollar e implementar, a través de la creación de un programa en Java, un sistema de administración escolar en un colegio específico, ahora con la integración de una base de datos MySQL. Este sistema, diseñado para optimizar la gestión académica, simplificar procesos administrativos y mejorar la accesibilidad a la información clave, se llevó a cabo con el propósito de proporcionar una herramienta efectiva que contribuya a la eficiencia operativa, la toma de decisiones informada y, en última instancia, elevar la calidad educativa en la institución.

### Específicos

Explorar fuentes académicas y literatura especializada para fundamentar teóricamente la necesidad y beneficios de un sistema de administración escolar.

Integrar los conceptos teóricos identificados en una base sólida que respalde la propuesta del sistema.

Priorizar las áreas de mejora según su impacto en la eficiencia y calidad académica.

Desarrollar un diseño detallado del sistema de administración escolar, utilizando la taxonomía de Bloom para estructurar la propuesta.

Crear el programa en Java según las especificaciones del diseño, siguiendo las mejores prácticas y estándares de desarrollo de software.

Implementar la integración con una base de datos MySQL para mejorar la eficiencia y persistencia de los datos.

Aplicar pruebas y evaluaciones rigurosas al sistema implementado, utilizando criterios verificables y medibles para asegurar su eficacia.

## Tipo de investigación

***Experimental***

Según Serrano (2011) “En la investigación de enfoque experimental el investigador manipula una o más variables de estudio, para controlar el aumento o disminución de esas variables y su efecto en las conductas observadas” **(p. 5)**.

En el caso del proyecto, se llevó a cabo un riguroso análisis e interpretación de los resultados obtenidos durante la ejecución del experimento. Este diseño de investigación implica la manipulación controlada de variables para observar y medir su impacto en un fenómeno específico, en este caso, la implementación del sistema de administración escolar con integración de una base de datos.

## Métodos de investigación

***Sintético***

Según Gonzales (2015) “El método sintético es un proceso de razonamiento que tiende a reconstruir un todo, a partir de los elementos distinguidos por el análisis; se trata en consecuencia de hacer una explosión metódica y breve” **(p. 26).**

Este trabajo garantiza que se pueda analizar y entender de manera integral los resultados de la investigación, permitiendo ofrecer una visión completa y fundamentada sobre el impacto del sistema de administración escolar en el contexto académico del colegio.

## Técnicas e instrumentos de investigación

### Implementación

Según Camino (1955) “En la literatura no se encuentra acuerdo sobre la definición de implementación. Es casi imposible desarrollar una definición precisa de implementación pues los investigadores desarrollan las definiciones de acuerdo a sus perspectivas del problema a resolver” **(p. 5).**

La técnica de implementación se refiere a la aplicación práctica y ejecución del sistema diseñado en el entorno escolar. Al seguir etapas y enfoques, se garantizará que el sistema de administración escolar con base de datos se introduzca y opere de manera exitosa en el entorno educativo.

# Resultados esperados

## Desarrollo de la propuesta

Cada uno de estos menús está diseñado para proporcionar al usuario opciones específicas para interactuar con la información de los alumnos y las materias, lo que permite una gestión detallada de los datos en el sistema de gestión escolar.

El menú “Alumno” permite al usuario interactuar con la información relacionada con los alumnos. Las opciones incluyen:

Ingresar nuevos alumnos.

Consultar información sobre un alumno por su ID.

Modificar datos de algún alumno.

Eliminar alumnos.

Imprimir la lista completa de alumnos.

El menú “Curso” se centra en la gestión de materias por curso. Ofrece dos opciones principales:

Modificar por ID de estudiante: Agregar, cambiar o eliminar materias especiales para un estudiante específico.

Modificar por curso: Agregar, cambiar o eliminar materias para un curso específico.

El menú “Materias” proporciona opciones para realizar acciones relacionadas con las materias en general. Incluye:

Modificar por ID de estudiante: Gestionar materias asociadas a un estudiante específico.

Modificar por curso: Gestionar materias asociadas a un curso específico.

## Fundamentación Teórica

La implementación de sistemas informáticos en la administración escolar es esencial para gestionar eficientemente información crítica como la matrícula, horarios y calificaciones.

Según Perry (2012) “Como cualquier lenguaje de programación, el lenguaje Java tiene su propia estructura, reglas de sintaxis y paradigma de programación” **(p. 3).**

A continuación, se detallan algunos de los conceptos relacionados con Java, los cuales se utilizaron en el programa:

**Clases y objetos**

Según Moltó (2007) “Java es un lenguaje de programación orientado a objetos, lo que significa que todo en Java es un objeto y se define en clases” **(p. 4).**

En el proyecto, se han definido varias clases como Alumno, Materia, AlumnoDAO, MateriaDAO, Connect, MenuCurso, ParteMaterias, subMenuAlumno, y submenus.

**Métodos y funciones**

Según Moltó (2007) “En Java, los métodos son bloques de código que se utilizan para realizar tareas específicas. Los métodos pueden tener parámetros y pueden devolver valores” **(p. 12).**

Se han definido varios métodos en las clases para realizar operaciones específicas como ingresar, modificar y eliminar datos.

**Encapsulamiento**

Según Moltó (2007) “Java soporta el encapsulamiento, que es el mecanismo de ocultar los detalles de implementación y exponer solo los aspectos necesarios de una clase. Esto se logra utilizando modificadores de acceso como public, private, y protected” **(p. 10).**

En el programa, se han encapsulado datos y métodos utilizando estos modificadores para proteger la integridad de los datos.

**Interfaces**

Según Moltó (2007) “Java admite la implementación de interfaces, que son similares a las clases, pero solo contienen definiciones de métodos y constantes” **(p. 54).**

Se han utilizado interfaces para definir un conjunto de métodos comunes que deben ser implementados por varias clases, facilitando la reutilización de código y la implementación de funcionalidades comunes.

**Entrada y salida**

Según Moltó (2007) “Java proporciona varias clases en el paquete java.util para realizar operaciones de entrada y salida **(p. 14).**

Se ha utilizado la clase Scanner para obtener datos de entrada del usuario desde la consola, permitiendo una interacción dinámica con el usuario.

**Manejo de excepciones**

Según Barnes (2003) “Java tiene un sistema de manejo de excepciones que permite manejar errores y excepciones durante la ejecución del programa” **(p. 396).**

En el programa, se ha implementado el manejo de excepciones para gestionar casos donde los usuarios ingresan datos incorrectos, garantizando que el programa pueda manejar errores de manera adecuada y no se detenga inesperadamente.

**Bucles y condiciones**

Según Barnes (2003) “Java proporciona estructuras de control como bucles while, do-while, y for, así como estructuras condicionales if, else if, y switch” **(p. 380).**

Estas estructuras se han utilizado en el programa para implementar la lógica de los menús y otras operaciones, permitiendo que el usuario navegue y realice acciones en el sistema de manera eficiente.

## Elaboración de la propuesta

### Pseudocódigo

Inicio

// Conectar a la base de datos

Conectar a base de datos usando Connect

Mientras el usuario no elija salir

Mostrar menú principal

Leer opción del usuario

Según opción seleccionada

Caso "Alumno":

Mostrar menú de Alumno

Leer opción del usuario

Según opción seleccionada

Caso "Ingresar nuevo alumno":

Leer datos del nuevo alumno

Llamar a AlumnoDAO para agregar el alumno a la base de datos

Mostrar mensaje de éxito

Caso "Consultar alumno por ID":

Leer ID del alumno

Llamar a AlumnoDAO para obtener información del alumno

Mostrar información del alumno

Caso "Modificar datos de alumno":

Leer ID del alumno

Llamar a AlumnoDAO para obtener información del alumno

Leer nuevos datos del alumno

Llamar a AlumnoDAO para actualizar el alumno en la base de datos

Mostrar mensaje de éxito

Caso "Eliminar alumno":

Leer ID del alumno

Llamar a AlumnoDAO para eliminar el alumno de la base de datos

Mostrar mensaje de éxito

Caso "Imprimir lista completa de alumnos":

Llamar a AlumnoDAO para obtener la lista completa de alumnos

Mostrar lista de alumnos

Caso "Curso":

Mostrar menú de Curso

Leer opción del usuario

Según opción seleccionada

Caso "Modificar por ID de estudiante":

Leer ID del estudiante

Llamar a MateriaDAO para obtener materias del estudiante

Leer nuevas materias

Llamar a MateriaDAO para actualizar materias del estudiante

Mostrar mensaje de éxito

Caso "Modificar por curso":

Leer ID del curso

Llamar a MateriaDAO para obtener materias del curso

Leer nuevas materias

Llamar a MateriaDAO para actualizar materias del curso

Mostrar mensaje de éxito

Caso "Materias":

Mostrar menú de Materias

Leer opción del usuario

Según opción seleccionada

Caso "Modificar por ID de estudiante":

Leer ID del estudiante

Llamar a MateriaDAO para obtener materias del estudiante

Leer nuevas materias

Llamar a MateriaDAO para actualizar materias del estudiante

Mostrar mensaje de éxito

Caso "Modificar por curso":

Leer ID del curso

Llamar a MateriaDAO para obtener materias del curso

Leer nuevas materias

Llamar a MateriaDAO para actualizar materias del curso

Mostrar mensaje de éxito

Caso "Salir":

Mostrar mensaje de despedida

Cerrar conexión a la base de datos

Terminar programa

Fin Según

Fin Mientras

Fin

### Código

Programa en Java. (**Anexo 2).**

### Capturas de la correcta ejecución

Programa impreso en pantalla y con los datos ingresados por teclado. **(Anexo 3).**

## Conclusiones

El sistema desarrollado permite una gestión integral de los datos escolares, abarcando tanto la información de los alumnos como de las materias. Esto facilita el seguimiento y la administración de los estudiantes y sus respectivas materias de manera eficiente.

Los menús implementados proporcionan una interfaz intuitiva que permite al usuario interactuar con el sistema de forma fácil y rápida. La estructura de menús ayuda a organizar las funciones del programa y simplifica la navegación entre diferentes opciones.

El uso de clases, métodos y encapsulamiento en Java demuestra una adecuada aplicación de los conceptos de programación orientada a objetos. Esto contribuye a una estructura de código más organizada y mantenible.

La implementación de manejo de excepciones asegura que el programa pueda manejar errores y situaciones inesperadas sin interrumpir su funcionamiento. Esto mejora la robustez y la fiabilidad del sistema.

La integración con la base de datos MySQL permite almacenar y recuperar datos de manera persistente. El uso de DAO (Data Access Object) para la gestión de datos refuerza la separación entre la lógica de negocio y el acceso a datos.

El uso de la clase Scanner para la entrada de datos y la salida de información demuestra una correcta implementación de técnicas de entrada y salida en Java.

## Recomendaciones

Considerar la optimización de consultas a la base de datos para mejorar el rendimiento del sistema, especialmente en escenarios con grandes volúmenes de datos.

Implementar una validación más exhaustiva de los datos ingresados por el usuario para evitar errores y asegurar la integridad de los datos en la base de datos.

Aunque el sistema actual es funcional, se podría mejorar la interfaz de usuario mediante el uso de una interfaz gráfica (GUI) para ofrecer una experiencia más amigable y visualmente atractiva.

Realizar pruebas adicionales en diferentes escenarios y con diferentes conjuntos de datos para asegurar que el sistema maneje correctamente todos los casos posibles.

Asegurarse de que toda la documentación del sistema esté actualizada y sea detallada. Esto incluye la documentación del código, manuales de usuario y guías para la instalación y mantenimiento del sistema.

Mejorar el manejo de errores mediante la implementación de logs para registrar eventos importantes y errores, lo que facilitará la depuración y el mantenimiento del sistema.

Implementar medidas de seguridad adicionales para proteger los datos almacenados en la base de datos, tales como cifrado de datos y control de acceso.

Mantener el sistema actualizado con nuevas tecnologías y técnicas para asegurar su relevancia y eficiencia en el tiempo.

## Link del Repositorio con avances del programa

Link que lleva a GitHub para descargar el programa. (**Anexo 4).**

# Referencias

Serrano, A. A. (2011). *Métodos de Investigación de Enfoque Experimental.* Obtenido de https://d1wqtxts1xzle7.cloudfront.net/55568285/Experimental-libre.pdf?1516242137=&response-content-disposition=inline%3B+filename%3DMETODOS\_DE\_INVESTIGACION\_DE\_ENFOQUE\_EXPE.pdf&Expires=1706069099&Signature=BSfVOAY5o7fHRSeLvYpkd4HJGxlZCnIKBWCWfqBQo~ebqqi7h

González, E. L. (2015). *El Método Científico.* Obtenido de https://www.ucm.es/data/cont/docs/107-2016-02-17-El%20M%C3%A9todo%20Cient%C3%ADfico.pdf

Camino, J. R. (Diciembre de 1995). *LA IMPLEMENTACION: UN FENOMENO ORGANIZATIVO MULTIDIMENSIONAL.* Obtenido de https://core.ac.uk/download/pdf/30041809.pdf

Perry, S. (03 de diciembre de 2012). *Conceptos básicos del lenguaje Java.* Obtenido de http://www.epet3.edu.ar/pampint/file/Tpampin1696.pdf

Barnes, D. J. (2003). *Programación orientada a objetos con Java.* Argentina: Miguel Martín-Romo.

Moltó, G. (2007). *Conceptos de Java para Estructuras de Datos.* Obtenido de https://d1wqtxts1xzle7.cloudfront.net/44511254/EDA\_Tema\_1\_gmolto-libre.pdf?1460054656=&response-content-disposition=inline%3B+filename%3DEDA\_Tema\_1\_gmolto.pdf&Expires=1708401581&Signature=OuqhWDdaXoQ91jUjP-7KG3N4MRz2qoplI0IgiLgzaGqG8c8SwcWEZ2BUcsmpOUYiIJ2

## Anexo 1 Esquema del problema

## Formulación y Definición del Problema

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Causas** | | **Definición del Problema** | **Consecuencias** | |
| * Falta de sistema centralizado. * Procesos manuales. * Desorganización. * Limitada accesibilidad | | Ineficiencia en la gestión académica debido a la falta de un sistema centralizado | * Ineficiencia y mayor propensión a errores en la gestión de información. * Desafíos en la ubicación y uso eficiente de información clave. * Dificultades en la toma de decisiones informadas. | |
|  | **Indicadores** | | |  |
| * Número de plataformas no integradas para la gestión de información académica. * Tasa de error en la administración de datos académicos. * Número de incidentes relacionados con la falta de estructura en la gestión de alumnos y materias. * Porcentaje de datos académicos restringidos o no accesibles. | | |

## 

## Anexo 2

## Código del programa

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany</groupId>

<artifactId>ProyectoEstudiantes</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>21</maven.compiler.source>

<maven.compiler.target>21</maven.compiler.target>

<exec.mainClass>com.mycompany.partematerias.ParteMaterias</exec.mainClass>

</properties>

<name>ProyectoEstudiantes</name>

</project>

**public class Alumno** {

private int Id\_Alumno;

private String Nombre;

private String Apellido;

private String Cedula;

private int Curso;

private String Contacto;

public Alumno(){}

public Alumno(int Id\_Alumno){

this.Id\_Alumno = Id\_Alumno;

}

// Constructor para cedula

public Alumno(String Cedula){

this.Cedula = Cedula;

}

public Alumno(String Nombre, String Apellido, String Cedula, int Curso, String Contacto){

this.Nombre = Nombre;

this.Apellido = Apellido;

this.Cedula = Cedula;

this.Curso = Curso;

this.Contacto = Contacto;

}

public Alumno(int Id\_Alumno, String Nombre, String Apellido,String Cedula, int Curso, String Contacto){

this.Id\_Alumno = Id\_Alumno;

this.Nombre = Nombre;

this.Apellido = Apellido;

this.Cedula = Cedula;

this.Curso = Curso;

this.Contacto = Contacto;

}

public int getId\_Alumno() {

return Id\_Alumno;

}

public void setId\_Alumno(int Id\_Alumno) {

this.Id\_Alumno = Id\_Alumno;

}

public String getNombre() {

return Nombre;

}

public void setNombre(String Nombre) {

this.Nombre = Nombre;

}

public String getApellido() {

return Apellido;

}

public void setApellido(String Apellido) {

this.Apellido = Apellido;

}

public String getCedula() {

return Cedula;

}

public void setCedula(String Cedula) {

this.Cedula = Cedula;

}

public int getCurso() {

return Curso;

}

public void setCurso(int Curso) {

this.Curso = Curso;

}

public String getContacto() {

return Contacto;

}

public void setContacto(String Contacto) {

this.Contacto = Contacto;

}

@Override

public String toString() {

return "Alumno{" +

"Id\_Alumno=" + Id\_Alumno +

", nombre='" + Nombre + '\'' +

", apellido='" + Apellido + '\'' +

", cedula='" + Cedula + '\'' +

", Curso='" + Curso + '\'' +

", Contacto='" + Contacto + '\'' +

'}';

}

}

import static com.mycompany.partematerias.Connect.getConnection;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

**public class AlumnoDAO** {

public List<Alumno> listarAlumnos(){

List<Alumno> alumnos = new ArrayList<>();

// Trabajo con clase de conexion a BD

PreparedStatement ps;

ResultSet rs;

Connection conect = getConnection();

String sql = "SELECT \* FROM alumno ORDER BY id;";

try{

ps = conect.prepareStatement(sql);

rs = ps.executeQuery();

while (rs.next()){

var alumno = new Alumno();

alumno.setId\_Alumno(rs.getInt("id"));

alumno.setNombre(rs.getString("Nombre"));

alumno.setApellido(rs.getString("Apellido"));

alumno.setCedula(rs.getString("Cedula"));

alumno.setCurso(rs.getInt("Curso"));

alumno.setContacto(rs.getString("Contacto"));

alumnos.add(alumno);

}

}

catch (SQLException e){

System.out.println("Ocurrio un error al listar los alumnos: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return alumnos;

}

public boolean findById(Alumno alumno) {

PreparedStatement ps;

ResultSet rs;

Connection conect = getConnection();

String sql = "SELECT \* FROM alumno WHERE id= ?;";

try{

ps = conect.prepareStatement(sql);

ps.setInt(1, alumno.getId\_Alumno());

rs = ps.executeQuery();

while (rs.next()){

alumno.setNombre(rs.getString("Nombre"));

alumno.setApellido(rs.getString("Apellido"));

alumno.setCedula(rs.getString("Cedula"));

alumno.setCurso(rs.getInt("Curso"));

alumno.setContacto(rs.getString("Contacto"));

return true;

}

}

catch (SQLException e){

System.out.println("Ocurrio un error al buscar alumno: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return false;

}

// Buscar alumno por cedula

public boolean findByCedula(Alumno alumno) {

PreparedStatement ps;

ResultSet rs;

Connection conect = getConnection();

String sql = "SELECT \* FROM alumno WHERE Cedula = ?;";

try{

ps = conect.prepareStatement(sql);

ps.setString(1, alumno.getCedula());

rs = ps.executeQuery();

while (rs.next()){

alumno.setNombre(rs.getString("Nombre"));

alumno.setApellido(rs.getString("Apellido"));

alumno.setCedula(rs.getString("Cedula"));

alumno.setCurso(rs.getInt("Curso"));

alumno.setContacto(rs.getString("Contacto"));

return true;

}

}

catch (SQLException e){

System.out.println("Ocurrio un error al buscar alumno: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return false;

}

public boolean agregarAlumno(Alumno alumno){

PreparedStatement ps;

Connection conect = getConnection();

String sql = "INSERT INTO alumno(Nombre, Apellido, Cedula, Curso, Contacto) " +

"VALUES(?, ?, ?, ?, ?);";

try{

ps = conect.prepareStatement(sql);

ps.setString(1, alumno.getNombre());

ps.setString(2, alumno.getApellido());

ps.setString(3, alumno.getCedula());

ps.setInt(4, alumno.getCurso());

ps.setString(5, alumno.getContacto());

ps.execute();

return true;

}

catch (SQLException e){

System.out.println("Ocurrio un error al agregar alumno: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return false;

}

public boolean actualizarAlumno(Alumno alumno){

PreparedStatement ps;

Connection conect = getConnection();

String sql = "UPDATE alumno SET nombre=?, apellido=?, cedula=?, Curso=?, Contacto=? " +

"WHERE id=?;";

try{

ps = conect.prepareStatement(sql);

ps.setString(1, alumno.getNombre());

ps.setString(2, alumno.getApellido());

ps.setString(3, alumno.getCedula());

ps.setInt(4, alumno.getCurso());

ps.setString(5, alumno.getContacto());

ps.setInt(6, alumno.getId\_Alumno());

ps.execute();

return true;

}

catch (SQLException e){

System.out.println("Ocurrio un error al actualizar alumno: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return false;

}

public boolean eliminarAlumno(Alumno alumno){

PreparedStatement ps;

Connection conect = getConnection();

String sql = "DELETE FROM alumno WHERE id = ?;";

try{

ps = conect.prepareStatement(sql);

ps.setInt(1, alumno.getId\_Alumno());

ps.execute();

return true;

}

catch (SQLException e){

System.out.println("Ocurrio un error al eliminar alumno: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return false;

}

//Eliminar Alumno por cedula

public boolean eliminarAlumnoCedula(Alumno alumno){

PreparedStatement ps;

Connection conect = getConnection();

String sql = "DELETE FROM alumno WHERE Cedula = ?;";

try{

ps = conect.prepareStatement(sql);

ps.setString(01, alumno.getCedula());

ps.execute();

return true;

}

catch (SQLException e){

System.out.println("Ocurrio un error al eliminar alumno: " + e.getMessage());

}

finally {

try{

conect.close();

}

catch (Exception e){

System.out.println("Error al cerrar conexion " + e.getMessage());

}

}

return false;

}

}

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

**public class Connect** {

public static Connection getConnection(){

Connection connection = null;

// definir datos para crear conexion

var database = "estudiantes\_app";

var url = "jdbc:mysql://localhost:3306/" + database;

var username = "root";

var password = "";

// Cargar clase del driver mysql en memoria

try{

Class.forName("com.mysql.cj.jdbc.Driver");

connection = DriverManager.getConnection(url, username, password);

}

//catch(Exception e){

catch(ClassNotFoundException | SQLException e){

System.out.println("Error en conexion a BD: " + e.getMessage());

}

return connection;

}

}

**public class Materia** {

private int id;

private String alumno;

private String materia;

private int numeroHoras;

public Materia(){}

public Materia(int id){

this.id = id;

}

public Materia(int num, String alum, String mat, int numH){

this.id=num;

this.alumno=alum;

this.materia=mat;

this.numeroHoras=numH;

}

public Materia(int num, String mat, int numH){

this.id=num;

this.materia=mat;

this.numeroHoras=numH;

}

public Materia(String mat, int numH){

this.materia=mat;

this.numeroHoras=numH;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getAlumno() {

return alumno;

}

public void setAlumno(String alumno) {

this.alumno = alumno;

}

public String getMateria() {

return materia;

}

public void setMateria(String materia) {

this.materia = materia;

}

public int getNumeroHoras() {

return numeroHoras;

}

public void setNumeroHoras(int numeroHoras) {

this.numeroHoras = numeroHoras;

}

@Override

public String toString() {

return "Materia{" + "id=" + id + ", alumno=" + alumno + ", materia=" + materia + ", numeroHoras=" + numeroHoras + '}';

}

}

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

**public class MateriaDAO** {

public void insertarMateria(int alumnoId, String materia, int horas) {

try (Connection conn = Connect.getConnection()) {

String sql = "INSERT INTO MateriaEsp (alumno\_id, materia, horas) VALUES (?, ?, ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, alumnoId);

stmt.setString(2, materia);

stmt.setInt(3, horas);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void eliminarMateria(int alumnoId, String materia) {

try (Connection conn = Connect.getConnection()) {

String sql = "DELETE FROM MateriaEsp WHERE alumno\_id = ? AND materia = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, alumnoId);

stmt.setString(2, materia);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public List<Materia> obtenerMateriasPorAlumno(int alumnoId) {

List<Materia> resultados = new ArrayList<>();

String sql = "SELECT A.nombre AS nombre\_alumno, M.materia AS nombre\_materia, ME.materia AS nombre\_materiaesp " +

"FROM Alumno A " +

"JOIN Materia M ON M.curso\_id = A.id\_curso " +

"JOIN MateriaEsp ME ON ME.id\_materia = A.id " +

"WHERE A.id = ?";

try (Connection conn = Connect.getConnection();

PreparedStatement stmt = conn.prepareStatement(sql)) {

stmt.setInt(1, alumnoId);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

String nombreAlumno = rs.getString("nombre\_alumno");

String nombreMateria = rs.getString("nombre\_materia");

String nombreMateriaEsp = rs.getString("nombre\_materiaesp");

resultados.add(new Materia(rs.getInt("id"), rs.getString("nombre"), rs.getString("materia"), rs.getInt("horas")));

}

} catch (SQLException e) {

e.printStackTrace();

}

return resultados;

}

public void insertarMateriaPorCurso(int cursoId, String materia, int horas) {

try (Connection conn = Connect.getConnection()) {

// Insertar la materia para todos los alumnos del curso

String sql = "INSERT INTO Materia (curso\_id, materia, horas) " +

"SELECT id\_curso, ?, ? FROM Alumnos WHERE curso\_id = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, materia);

stmt.setInt(2, horas);

stmt.setInt(3, cursoId);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void eliminarMateriaPorCurso(int cursoId, String materia) {

try (Connection conn = Connect.getConnection()) {

// Eliminar la materia para todos los alumnos del curso

String sql = "DELETE FROM Materia WHERE materia = ? AND curso\_id IN " +

"(SELECT id\_curso FROM Alumnos WHERE curso\_id = ?)";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, materia);

stmt.setInt(2, cursoId);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public List<Materia> obtenerMateriasPorCurso(int cursoId) {

List<Materia> materias = new ArrayList<>();

try (Connection conn = Connect.getConnection()) {

// Obtener todas las materias para los alumnos del curso

String sql = "SELECT M.curso\_id, M.materia, M.horas FROM Materias M " +

"JOIN Alumnos A ON M.curso\_id = A.id\_curso WHERE A.id\_curso = ?";

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setInt(1, cursoId);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

materias.add(new Materia(rs.getInt("curso\_id"), rs.getString("nombre"), rs.getInt("horas")));

}

} catch (SQLException e) {

e.printStackTrace();

}

return materias;

}

}

import java.util.Scanner;

**public class MenuCurso** {

Scanner sc = new Scanner(System.in);

public void MenuCurso() {

submenus sm = new submenus();

int a = 0;

while (a != 3) {

System.out.println("----Menu de Materias----");

System.out.println("1. Modificar por ID de Alumno");

System.out.println("2. Modificar por curso");

System.out.println("3. Salir");

while (!sc.hasNextInt()) {

System.out.println("Ingresar un número del 1 al 3");

sc.next();

}

a = sc.nextInt();

if (a < 1 || a > 3) {

System.out.println("Ingrese una opcion del 1 al 3");

} else if (a == 1 || a == 2) {

sm.submenus(a);

}

}

}

}

import java.util.Scanner;

**public class ParteMaterias** {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

MenuCurso menu = new MenuCurso();

subMenuAlumno subMA = new subMenuAlumno();

int op = 0;

do {

System.out.println("-----Registro Colegio-----");

System.out.println("1. Alumnos");

System.out.println("2. Materias");

System.out.println("3. Salir");

op = sc.nextInt();

if (op == 1) {

subMA.subMenuAlumno();

} else if (op == 2) {

menu.MenuCurso();

}

} while (op != 3);

}

}

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

**public class subMenuAlumno** {

public void subMenuAlumno(){

Scanner consola = new Scanner(System.in);

var salir = false;

var alumnoDAO = new AlumnoDAO();

// MUESTRA MENU

while (!salir){

try{

muestramenu();

salir = ejecutarOpciones(consola, alumnoDAO);

}

catch (Exception e){

System.out.println();

System.out.println("Error en ejecución: " + e.getMessage());

System.out.println();

}

}

}

private static void muestramenu(){

System.out.println();

System.out.println("""

\*\*\*\*\* SISTEMA DE ALUMNOS \*\*\*\*\*

1. Listar

2. Buscar alumno por Id\_Alumno

3. Buscar alumno por cedula

4. Agregar

5. Actualizar

6. Eliminar por Id\_Alumno

7. Eliminar alumno por cedula

8. Salir

""");

System.out.print("Por favor, ingrese una opción: ");

System.out.println();

}

private static boolean ejecutarOpciones(Scanner consola, AlumnoDAO alumnoDAO){

var opcion = Integer.parseInt(consola.nextLine());

System.out.println();

boolean salir = false;

// Validar opciones ingresadas

switch (opcion){

case 1 -> { // Listar alumno

System.out.println();

System.out.println(".:: Lista de alumno ::.");

var alumnos = alumnoDAO.listarAlumnos();

alumnos.forEach(System.out::println);

}

case 2 -> { // Buscar por ID

System.out.println("Ingrese ID de alumno a buscar: ");

var Id\_Alumno = Integer.parseInt(consola.nextLine());

var alumno = new Alumno(Id\_Alumno);

var existeAlumno = alumnoDAO.findById(alumno);

if(existeAlumno)

System.out.println("Alumno encontrado: " + alumno);

else

System.out.println("No se encontro alumno con ID : " + alumno.getId\_Alumno());

}

case 3 -> { // Buscar por cedula

System.out.println("Ingrese cedula de alumno a buscar: ");

var cedula = consola.nextLine();

var alumno = new Alumno(cedula);

var existeAlumno = alumnoDAO.findByCedula(alumno);

if(existeAlumno)

System.out.println("Alumno encontrado: " + alumno);

else

System.out.println("No se encontro alumno con ID : " + alumno.getCedula());

}

case 4 -> { // Agregar alumno

Scanner scanner =new Scanner(System.in);

String nombre1=null;

String apellido1=null;

String cedula1=null;

String contacto1=null;

int curso1=0;

System.out.println(".:: Agregar Alumno ::.");

// validacion prueba

int sw =0;

do{

System.out.print("Nombre: ");

nombre1 =scanner.nextLine();

//var nombre1 = consola.nextLine();

Pattern pattern = Pattern.compile ("^[A-Za-zÀ-ÖØ-öø-ÿ'‘’“”„“”\\s-]+$"); // PARA CARACTERES A-Z a-z

Matcher matcher = pattern.matcher(nombre1);

if (matcher.matches())

{

var nombre = nombre1;

sw=1;

}

else

{

System.out.println("El nombre no es valido");

}

}while(sw!=1);

//fin validacion nombre

//validar apellido

int sw1 =0;

do{

System.out.print("Apellido: ");

apellido1 =scanner.nextLine();

Pattern pattern = Pattern.compile ("^[A-Za-zÀ-ÖØ-öø-ÿ'‘’“”„“”\\s-]+$"); // PARA CARACTERES A-Z a-z

Matcher matcher = pattern.matcher(apellido1);

if (matcher.matches())

{

var apellido = apellido1;

sw1=1;

}

else

{

System.out.println("El apellido no es valido");

}

}while(sw1!=1);

//Fin valida apellido

//Valida cedula

int sw4=0;

do

{

int numero=0,suma=0,resultado=0, num=0;

sw=0;

System.out.println("Ingrese la cedula del alumno");

cedula1=scanner.nextLine();

num=cedula1.length();

System.out.println("longitud: "+num);

Pattern pattern = Pattern.compile ("^[0-9]+$"); //SOLO CARACTERES NUMEROS

Matcher matcher = pattern.matcher(cedula1);

if (matcher.matches()&& num==10)

{

for (int i=0; i< cedula1.length();i++)

{

numero=Integer.parseInt(String.valueOf(cedula1.charAt(i)));

if (i%2==0)

{

numero=numero\*2;

if (numero>9)

{

numero=numero-9;

}

}

suma=suma+numero;

}

if (suma%10 !=0)

{

resultado=10-(suma%10);

if (resultado == numero)

{

System.out.println("Su cedula "+cedula1+" es ecuatoriana ");

sw=1;

}

else

{

System.out.println("Su cedula "+cedula1+" no es ecuatoriana ");

}

}

else

{

System.out.println("Su cedula "+cedula1+" es ecuatoriana ");

sw4=1;

}

}

}while(sw4!=1);

//Fin valida cedula

//Valida Curso

int sw3 =0;

do{

System.out.print("Curso: ");

curso1 =scanner.nextInt();

Pattern pattern = Pattern.compile ("^[0-9]+$"); // Solo caracteres numeros

if (scanner.hasNextInt()) {

curso1 = scanner.nextInt();

sw3 = 1; // Entrada válida, salir del bucle

} else {

System.out.println("El curso no es válido");

scanner.next(); // Limpia la entrada inválida

}

}while(sw3!=1);

//fin valida curso

//valida Contacto

int sw2 =0;

do{

System.out.print("Contacto: ");

contacto1 =scanner.nextLine();

Pattern pattern = Pattern.compile ("^[A-Za-zÀ-ÖØ-öø-ÿ'‘’“”„“”\\s-]+$"); // PARA CARACTERES A-Z a-z

Matcher matcher = pattern.matcher(contacto1);

if (matcher.matches())

{

var contacto = contacto1;

sw2=1;

}

else

{

System.out.println("El contacto no es valido");

}

}while(sw2!=1);

//fin valida contacto

var nuevoAlumno = new Alumno(nombre1, apellido1,cedula1, curso1, contacto1);

var agregaalumno = alumnoDAO.agregarAlumno(nuevoAlumno);

if(agregaalumno)

System.out.println("Alumno agregado exitosamente: " + nuevoAlumno);

else

System.out.println("No se pudo agregar alumno : " + nuevoAlumno);

}

case 5 -> { // Actualizar alumno

System.out.println(".:: Actualizar Alumno ::.");

System.out.println("Ingrese ID de alumno a actualizar: ");

var idalumno = Integer.parseInt(consola.nextLine());

Scanner scanner =new Scanner(System.in);

String nombre1=null;

String apellido1=null;

String cedula1=null;

String contacto1=null;

int curso1=0;

System.out.println(".:: Actualizar Alumno ::.");

// validacion prueba

int sw =0;

do{

System.out.print("Nombre: ");

nombre1 =scanner.nextLine();

//var nombre1 = consola.nextLine();

Pattern pattern = Pattern.compile ("^[A-Za-zÀ-ÖØ-öø-ÿ'‘’“”„“”\\s-]+$"); // PARA CARACTERES A-Z a-z

Matcher matcher = pattern.matcher(nombre1);

if (matcher.matches())

{

var nombre = nombre1;

sw=1;

}

else

{

System.out.println("El nombre no es valido");

}

}while(sw!=1);

//fin validacion nombre

//validar apellido

int sw1 =0;

do{

System.out.print("Apellido: ");

apellido1 =scanner.nextLine();

Pattern pattern = Pattern.compile ("^[A-Za-zÀ-ÖØ-öø-ÿ'‘’“”„“”\\s-]+$"); // PARA CARACTERES A-Z a-z

Matcher matcher = pattern.matcher(apellido1);

if (matcher.matches())

{

var apellido = apellido1;

sw1=1;

}

else

{

System.out.println("El apellido no es valido");

}

}while(sw1!=1);

//Fin valida apellido

//Valida cedula

int sw4=0;

do

{

int numero=0,suma=0,resultado=0, num=0;

sw=0;

System.out.println("Ingrese la cedula del alumno");

cedula1=scanner.nextLine();

num=cedula1.length();

System.out.println("longitud: "+num);

Pattern pattern = Pattern.compile ("^[0-9]+$"); //SOLO CARACTERES NUMEROS

Matcher matcher = pattern.matcher(cedula1);

if (matcher.matches()&& num==10)

{

for (int i=0; i< cedula1.length();i++)

{

numero=Integer.parseInt(String.valueOf(cedula1.charAt(i)));

if (i%2==0)

{

numero=numero\*2;

if (numero>9)

{

numero=numero-9;

}

}

suma=suma+numero;

}

if (suma%10 !=0)

{

resultado=10-(suma%10);

if (resultado == numero)

{

System.out.println("Su cedula "+cedula1+" es ecuatoriana ");

sw=1;

}

else

{

System.out.println("Su cedula "+cedula1+" no es ecuatoriana ");

}

}

else

{

System.out.println("Su cedula "+cedula1+" es ecuatoriana ");

sw4=1;

}

}

}while(sw4!=1);

//Fin valida cedula

//Valida Curso

int sw3 =0;

do{

System.out.print("Curso: ");

curso1 =scanner.nextInt();

Pattern pattern = Pattern.compile ("^[0-9]+$"); // Solo caracteres numeros

if (scanner.hasNextInt()) {

curso1 = scanner.nextInt();

sw3 = 1; // Entrada válida, salir del bucle

} else {

System.out.println("El curso no es válido");

scanner.next(); // Limpia la entrada inválida

}

}while(sw3!=1);

//fin valida curso

//valida Contacto

int sw2 =0;

do{

System.out.print("Contacto: ");

contacto1 =scanner.nextLine();

Pattern pattern = Pattern.compile ("^[A-Za-zÀ-ÖØ-öø-ÿ'‘’“”„“”\\s-]+$"); // PARA CARACTERES A-Z a-z

Matcher matcher = pattern.matcher(contacto1);

if (matcher.matches())

{

var contacto = contacto1;

sw2=1;

}

else

{

System.out.println("El contacto no es valido");

}

}while(sw2!=1);

//fin valida contacto

var editaAlumno = new Alumno(idalumno, nombre1, apellido1, cedula1,curso1, contacto1);

var actualizado = alumnoDAO.actualizarAlumno(editaAlumno);

if(actualizado)

System.out.println("Alumno actualizado exitosamente: " + editaAlumno);

else

System.out.println("No se pudo actualizar alumno : " + editaAlumno);

}

case 6 -> { // Eliminar alumno

System.out.println(".:: Eliminar alumno ::.");

System.out.println("Ingrese ID de alumno a eliminar: ");

var idAlumno = Integer.parseInt(consola.nextLine());

var eliminaAlumno = new Alumno(idAlumno);

var eliminado = alumnoDAO.eliminarAlumno(eliminaAlumno);

if(eliminado)

System.out.println("Alumno eliminado exitosamente: " + eliminaAlumno);

else

System.out.println("No se pudo eliminar alumno : " + eliminaAlumno);

}

case 7 -> { // Eliminar alumno por cedula

System.out.println(".:: Eliminar alumno por cedula ::.");

System.out.println("Ingrese cedula del alumno a eliminar: ");

System.out.print("Cedula: ");

var cedula = consola.nextLine();

var eliminaAlumno = new Alumno(cedula);

var eliminado = alumnoDAO.eliminarAlumnoCedula(eliminaAlumno);

if(eliminado)

System.out.println("Alumno eliminado exitosamente: " + eliminaAlumno);

else

System.out.println("No se pudo eliminar alumno : " + eliminaAlumno);

}

case 8 -> { // Salir de la app

System.out.println("Gracias, hasta pronto!!!");

salir = true;

}

default -> System.out.println("Opcion ingresada no existe, por favor ingrese nuevamente.");

}

return salir;

}

}

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

import java.util.Scanner;

**public class submenus** {

Scanner sc = new Scanner(System.in);

public submenus() {}

public void submenus(int a) {

int id;

int cr;

if (a == 1) {

System.out.println("Ingrese el ID del Alumno");

while (!sc.hasNextInt()) {

System.out.println("Ingresar un ID");

sc.next();

}

id = sc.nextInt();

try {

if (buscarAlumnoPorID(id)) {

porID(id);

} else {

System.out.println("No se encontro al alumno con el ID proporcionado.");

}

} catch (SQLException e) {

e.printStackTrace();

}

} else {

System.out.println("Ingrese el curso del 1 al 3");

while (true) {

while (!sc.hasNextInt()) {

System.out.println("Ingresar un curso del 1 al 3");

sc.next();

}

cr = sc.nextInt();

if(cr>=1 && cr<=3) {

break; // Sale del bucle si el número está dentro del rango

}else{

System.out.println("Ingresar un curso del 1 al 3");

}

}

porCurso(cr);

}

}

private boolean buscarAlumnoPorID(int id) throws SQLException {

Connection con = Connect.getConnection();

String query = "SELECT COUNT(\*) FROM alumnos WHERE id = ?";

PreparedStatement ps = con.prepareStatement(query);

ps.setInt(1, id);

ResultSet rs = ps.executeQuery();

if (rs.next() && rs.getInt(1) > 0) {

return true;

}

return false;

}

void porID(int a) {

MateriaDAO materiaDAO = new MateriaDAO();

int op = 0;

String mat;

int h;

do {

System.out.println("--Materias del Alumno--");

System.out.println("1. Ingresar");

System.out.println("2. Cambiar");

System.out.println("3. Eliminar");

System.out.println("4. Consultar");

System.out.println("5. Salir");

while (!sc.hasNextInt()) {

System.out.println("Ingresar una opción del 1 al 5");

sc.next();

}

op = sc.nextInt();

sc.nextLine();

switch (op) {

case 1:

System.out.println("Ingrese la nueva materia:");

mat = sc.nextLine();

System.out.println("Ingrese el número de horas a cursar:");

while (!sc.hasNextInt()) {

System.out.println("Ingrese un número válido de horas:");

sc.next();

}

h = sc.nextInt();

materiaDAO.insertarMateria(a, mat, h);

break;

case 2:

System.out.println("Ingrese la materia a cambiar:");

mat = sc.nextLine();

materiaDAO.eliminarMateria(a, mat);

System.out.println("Ingrese la nueva materia:");

mat = sc.nextLine();

System.out.println("Ingrese el número de horas a cursar:");

while (!sc.hasNextInt()) {

System.out.println("Ingrese un número válido de horas:");

sc.next();

}

h = sc.nextInt();

materiaDAO.insertarMateria(a, mat, h);

break;

case 3:

System.out.println("Ingrese la materia a eliminar:");

mat = sc.nextLine();

materiaDAO.eliminarMateria(a, mat);

break;

case 4:

System.out.println("Materias del estudiante:");

List<Materia> materias = materiaDAO.obtenerMateriasPorAlumno(a);

for (Materia m : materias) {

System.out.println("ID alumno: "+m.getId() + "Alumno: " + m.getAlumno() + "Materia: " + m.getMateria() + ", Horas: " + m.getNumeroHoras());

}

break;

case 5:

System.out.println("Saliendo...");

break;

default:

System.out.println("Opción no válida, ingrese un número del 1 al 5.");

}

} while (op != 5);

}

void porCurso(int a) {

MateriaDAO materiaDAO = new MateriaDAO();

int op = 0;

String mat;

int h;

do {

System.out.println("--Materia por curso--");

System.out.println("1. Ingresar");

System.out.println("2. Cambiar");

System.out.println("3. Eliminar");

System.out.println("4. Consultar");

System.out.println("5. Salir");

while (!sc.hasNextInt()) {

System.out.println("Ingresar una opción del 1 al 5");

sc.next();

}

op = sc.nextInt();

sc.nextLine();

switch (op) {

case 1:

System.out.println("Ingrese la nueva materia:");

mat = sc.nextLine();

System.out.println("Ingrese el número de horas a cursar:");

while (!sc.hasNextInt()) {

System.out.println("Ingrese un número válido de horas:");

sc.next();

}

h = sc.nextInt();

// Aquí se necesita un método para insertar materias por curso en lugar de por alumno.

materiaDAO.insertarMateriaPorCurso(a, mat, h);

break;

case 2:

System.out.println("Ingrese la materia a cambiar:");

mat = sc.nextLine();

materiaDAO.eliminarMateriaPorCurso(a, mat);

System.out.println("Ingrese la nueva materia:");

mat = sc.nextLine();

System.out.println("Ingrese el número de horas a cursar:");

while (!sc.hasNextInt()) {

System.out.println("Ingrese un número válido de horas:");

sc.next();

}

h = sc.nextInt();

materiaDAO.insertarMateriaPorCurso(a, mat, h);

break;

case 3:

System.out.println("Ingrese la materia a eliminar:");

mat = sc.nextLine();

materiaDAO.eliminarMateriaPorCurso(a, mat);

break;

case 4:

System.out.println("Materias del curso:");

List<Materia> materias = materiaDAO.obtenerMateriasPorCurso(a);

for (Materia m : materias) {

System.out.println("Curso: " + m.getId() + "Materia: " + m.getMateria() + ", Horas: " + m.getNumeroHoras());

}

break;

case 5:

System.out.println("Saliendo...");

break;

default:

System.out.println("Opción no válida, ingrese un número del 1 al 5.");

}

} while (op != 5);

}

}

## Anexo 3

## Capturas de la correcta ejecución

El programa comienza con un menú que permite elegir entre operaciones relacionadas con Alumnos, Materias o Salir.

(CAPTURAS DE EJECUCIÓN)

## Anexo 4

## Link del Repositorio

<https://github.com/FelipeAMC01/Codigo-provisional.git>