**Proyecto de Gestión de Estudiantes en Java**

**Descripción General**

Este proyecto es una aplicación de consola desarrollada en Java que permite administrar información básica de estudiantes. Utiliza una base de datos MySQL para almacenar los datos y sigue una arquitectura MVC (Modelo-Vista-Controlador) para organizar el código y separar responsabilidades. El sistema permite realizar operaciones básicas CRUD: Crear, Leer, Actualizar y Eliminar registros de estudiantes.

**Objetivos**

* Implementar una aplicación Java conectada a una base de datos relacional.
* Aplicar conceptos de programación orientada a objetos y diseño modular.
* Demostrar el uso de JDBC para operaciones con bases de datos.
* Proporcionar una interfaz sencilla para la gestión de estudiantes mediante consola.

**Estructura del Proyecto y Componentes**

**1. Modelo (com.app.Model.Estudiante)**

* **Propósito:** Representar la entidad "Estudiante" en el sistema.
* **Atributos:**
  + id (int): Identificador único del estudiante.
  + nombre (String): Nombre completo del estudiante.
  + edad (int): Edad del estudiante.
  + carrera (String): Carrera o especialidad que estudia el estudiante.
* **Métodos:**
  + Getters y setters para cada atributo, que permiten obtener y modificar los valores.
  + Constructores para crear instancias de estudiante con o sin parámetros.

**2. Capa de Acceso a Datos (com.app.Dao)**

* **Clase ConexionBD:**
  + Encargada de establecer y cerrar la conexión con la base de datos MySQL usando JDBC.
  + Contiene métodos estáticos:
    - getConnection(): Retorna una conexión activa a la base de datos.
    - closeConnection(Connection connection): Cierra la conexión pasada como parámetro.
  + Configuración necesaria: URL de la base de datos, usuario y contraseña.
* **Clase EstudianteDAO:**
  + Realiza operaciones CRUD en la tabla estudiantes de la base de datos.
  + Métodos principales:
    - agregar(Estudiante e): Inserta un nuevo estudiante en la base de datos.
    - obtenerPorId(int id): Recupera un estudiante por su ID.
    - obtenerTodos(): Obtiene una lista con todos los estudiantes.
    - actualizar(Estudiante e): Actualiza los datos de un estudiante existente.
    - eliminar(int id): Elimina un estudiante según su ID.

**3. Controlador (com.app.Controller.EstudianteController)**

* **Función:** Actúa como intermediario entre la vista y el acceso a datos.
* Gestiona la lógica de negocio y la validación de datos antes de llamar al DAO.
* Métodos correspondientes a las operaciones CRUD que llaman a los métodos del DAO.
* Facilita la comunicación y el flujo de datos hacia y desde la interfaz de usuario.

**4. Vista (com.app.View.EstudianteView)**

* **Interfaz:** Basada en consola, sencilla y amigable.
* Muestra un menú principal con las opciones disponibles para el usuario.
* Solicita al usuario la información necesaria para cada operación (ejemplo: datos del estudiante a agregar).
* Muestra mensajes de éxito, error o resultados según la operación realizada.
* Controla el flujo del programa, llamando al controlador según la opción seleccionada.

**5. Clase Principal (com.app.Main)**

* Punto de entrada de la aplicación.
* Inicializa la vista y la pone en marcha para que el usuario interactúe.
* Contiene el método main(String[] args) que ejecuta el programa.

**Detalles Técnicos y Configuración**

**Base de Datos**

* Debe existir una base de datos MySQL creada previamente.
* Se requiere una tabla llamada estudiantes con la siguiente estructura:

| **Campo** | **Tipo** | **Descripción** |
| --- | --- | --- |
| id | INT | Llave primaria, auto-increment |
| nombre | VARCHAR(50) | Nombre del estudiante |
| edad | INT | Edad del estudiante |
| carrera | VARCHAR(50) | Carrera o especialidad |

* Se recomienda configurar usuario y contraseña en ConexionBD.java.

**Librerías**

* Se utiliza la librería mysql-connector-j-8.4.0.jar para conectar Java con MySQL.
* Esta librería debe estar ubicada en la carpeta lib/ y referenciada en el proyecto.

**Cómo Usar el Sistema**

1. **Iniciar la aplicación:** Ejecutar la clase Main.
2. **Menú principal:** Aparecerán las opciones para:
   * Agregar estudiante.
   * Listar todos los estudiantes.
   * Buscar estudiante por ID.
   * Actualizar datos de un estudiante.
   * Eliminar estudiante.
   * Salir del programa.
3. **Seleccionar opción:** Ingresar el número correspondiente a la opción deseada.
4. **Ingresar datos:** Según la opción, el sistema solicitará los datos necesarios, por ejemplo, al agregar un estudiante se pedirá nombre, edad y carrera.
5. **Visualizar resultados:** El sistema mostrará mensajes indicando éxito o fallo de la operación.
6. **Continuar o salir:** Luego de cada operación, volverás al menú principal para realizar otra acción o salir.

**Ejemplo de Flujo de Uso**

* El usuario inicia el programa y ve el menú.
* Selecciona "1" para agregar un estudiante.
* Ingresa el nombre, edad y carrera.
* El sistema guarda los datos en la base de datos y confirma la operación.
* Luego, selecciona "2" para listar todos los estudiantes y verifica que el nuevo estudiante aparece en la lista.
* Puede buscar, actualizar o eliminar estudiantes usando sus IDs.

**Beneficios y Aplicaciones**

* Facilita la gestión de información estudiantil para pequeños proyectos o prácticas.
* Sirve como base para desarrollar aplicaciones más complejas con interfaces gráficas o web.
* Permite entender la integración entre Java y bases de datos relacionales.
* Refuerza conceptos de arquitectura MVC y diseño orientado a objetos.

**Documentación Técnica**

La documentación completa del código, incluyendo la descripción detallada de clases, métodos y paquetes, está generada con Javadoc y se encuentra en la carpeta doc del proyecto.

Para acceder a la documentación, abre la carpeta:

\Acceso\_Datos\doc

Esta documentación contiene toda la información técnica necesaria para entender el funcionamiento interno de la aplicación.