Memoria Practica V Tema ORM

Que

Modifica el programa anterior para que utlitzant un ORM (HIBENATE per exemple) almacene a una Base de Dades

Para que

Este ejercicio ha resultado útil para aprender y practicar el funcionamiento de un ORM como Hibernate. También ha sido beneficioso para adquirir conocimientos sobre bases de datos, ya que fue necesario crearla y realizar un par de consultas en ella. La lógica del ORM en sí no resultó ser muy complicada. En mi experiencia, la dificultad de este programa radicó en la conexión con la base de datos y el tiempo de programación, dado que es un programa extenso. Además, invertí tiempo en la búsqueda y corrección de errores. En resumen, la actividad fue útil para comprender mejor el funcionamiento de un ORM, así como para aprender a crear sesiones, realizar consultas, entre otros aspectos.

Pseudocodigo

Package Curso

Alumno

```
Clase Alumno:
    Atributos:
        - id (entero)
        - nombre (cadena de caracteres)
        - apellidos (cadena de caracteres)
        - fechaNacimiento (cadena de caracteres)
        - nia (entero)

    Métodos:
        Constructor():
            // Constructor predeterminado sin parámetros

        Constructor(nombre, apellidos, fechaNacimiento, nia):
            // Constructor con parámetros para inicializar la clase
        getters y setters
        setFechaNacimiento(dia, mes, ano):
            Si dia < 32 y mes < 13 y ano > 0 y ano <= año actual:
                fecha = formatearFecha(dia, mes, ano)
```

```
                this.fechaNacimiento = fecha
            Sino:
                Mostrar mensaje de error "La fecha no se ha podido
modificar"
        toString():
            Devolver nia concatenado con " : " y el nombre completo del
alumno
```

Matricula

```
Clase Matricula:
    Atributos:
        - id (entero)
        - alumno (instancia de la clase Alumno)
        - modulo (instancia de la clase Modulo)
        - notas (cadena de caracteres)

    Métodos:
        Constructor():
            // Constructor predeterminado sin parámetros
        Constructor(alumno, modulo, notas):
            // Constructor con parámetros para inicializar la clase
        getters y setters
        getNotasList():
            Devolver lista de números obtenida del string de notas

        setNotasList(notasList):
            Convertir la lista de números a una cadena y asignarla a
'notas'

        addNota(notas):
            Obtener la lista actual de notas
            Crear una nueva lista con las notas antiguas y las nuevas
            Llamar a setNotasList con la nueva lista de notas

        mostrarNotas():
            Obtener la lista de notas
            Si la lista de notas es nula:
                Mostrar un mensaje de advertencia indicando que el alumno
no tiene notas en el módulo
            Sino:
                Mostrar las notas del alumno en el módulo, enumeradas
```

Modulo

```
Clase Modulo:
    Atributos:
        - id (entero)
        - nombre (cadena de caracteres)

    Métodos:
        Constructor():
            // Constructor predeterminado sin parámetros

        Constructor(nombre):
            // Constructor con parámetros para inicializar la clase
        getters y setters
        toString():
            Devolver "ID: " concatenado con el id y ", Nombre: "
concatenado con el nombre
```

Package gestor

Conexion

```
Clase Conexion:
    Atributos:
        - sessionFactory (instancia de SessionFactory)

    Métodos:
        Constructor (privado):
            Llamar a buildSessionFactory y asignar el resultado a
sessionFactory

        buildSessionFactory():
            Try:
                Crear la fábrica de sesiones a partir de la configuración
en hibernate.cfg.xml
                Devolver la fábrica de sesiones creada
            Catch Throwable ex:
                Imprimir el error al inicializar la SessionFactory
                Lanzar una ExceptionInInitializerError con el error

        getSessionFactory():
            Devolver la instancia de SessionFactory
```

gestorDB

```
Clase gestorDB:
    Métodos:
        addDB(objeto):
            Abrir una nueva sesión
            Iniciar una transacción
            Guardar el objeto en la base de datos
            Confirmar la transacción
            Cerrar la sesión
```

gsAlumnos

```
Clase gsAlumnos extiende gestorDB:
    Atributos:
        - rc (instancia de ReadClient)

    Métodos:
        alta():
            Pedir el nombre, apellidos, NIA y fecha de nacimiento del
alumno
            Crear una nueva instancia de Alumno con la información
proporcionada
            Llamar al método addDB de la clase base para agregar el alumno
a la base de datos
            Mostrar un mensaje de éxito

        baja():
            Pedir el NIA del alumno a eliminar
            Si el NIA es diferente de 0:
                Abrir una nueva sesión
                Iniciar una transacción
                Intentar:
                    Buscar el alumno por NIA
                    Si el alumno existe:
                        Crear una instancia de gsMatriculas y eliminar las
matrículas asociadas al NIA
                        Eliminar el alumno de la base de datos
                        Confirmar la transacción
                        Mostrar un mensaje de éxito
                    Sino:
                        Mostrar un mensaje de error indicando que no se
encontró el alumno
                Capturar Exception e:
```

```
                    Revertir la transacción
                    Mostrar un mensaje de error con la descripción del
error
                Finalmente:
                    Cerrar la sesión

        mostrarAlumnos():
            Abrir una nueva sesión
            Crear una consulta para obtener todos los alumnos
            Obtener la lista de alumnos
            Mostrar la información de cada alumno en la lista
            Cerrar la sesión

        pedirNia(exist):
            Mostrar un mensaje solicitando el NIA
            Si exist es verdadero:
                Mostrar un mensaje adicional indicando que 0 se puede usar
para cancelar
                Mostrar un mensaje de error si el NIA ya existe
(comprobarNia devuelve falso)
            Pedir un valor positivo para el NIA
            Repetir hasta que se ingrese un NIA válido

        buscarAlmNia(nia):
            Abrir una nueva sesión
            Intentar:
                Crear una consulta para obtener el alumno por NIA
                Ejecutar la consulta y devolver el resultado
            Capturar Exception e:
                Mostrar un mensaje de error indicando que no se pudo buscar
al alumno
            Finalmente:
                Cerrar la sesión

        comprobarNia(nia, exist):
            Abrir una nueva sesión
            Crear una consulta para obtener todos los NIA de la base de
datos
            Obtener la lista de NIA
            Cerrar la sesión
            Si exist es verdadero:
                Devolver verdadero si el NIA existe en la lista, falso si
no existe
            Si exist es falso:
```

```
                Devolver verdadero si el NIA NO existe en la lista, falso
si existe
```

gsModulos

```
Clase gsModulos extiende gestorDB:
    Atributos:
        - rc (instancia de ReadClient)

    Métodos:
        alta():
            Pedir el nombre del nuevo módulo
            Crear una nueva instancia de Modulo con el nombre proporcionado
            Llamar al método addDB de la clase base para agregar el módulo
a la base de datos
            Mostrar un mensaje de éxito

        baja():
            Pedir el ID del módulo a eliminar
            Si el ID es diferente de 0:
                Llamar al método eliminar con el ID proporcionado
                Mostrar un mensaje de éxito
            Sino:
                Mostrar un mensaje de advertencia indicando que se ha
cancelado la opción

        matricular():
            Crear instancias de gsAlumnos y gsMatriculas
            Pedir el NIA del alumno y el ID del módulo
            Si el NIA es diferente de 0:
                Buscar al alumno por NIA
                Buscar el módulo por ID
                Si el alumno existe y el módulo no está vacío:
                    Buscar una matrícula existente para el alumno en el
módulo
                    Si no hay matrícula existente:
                        Crear una nueva matrícula para el alumno en el
módulo y agregarla a la base de datos
                        Mostrar un mensaje de éxito
                    Sino:
                        Mostrar un mensaje de error indicando que el alumno
ya está matriculado

        mostrarModulos():
```

```
            Abrir una nueva sesión
            Crear una consulta para obtener todos los módulos
            Obtener la lista de módulos
            Mostrar la información de cada módulo en la lista
            Cerrar la sesión

        pedirId():
            Pedir un valor positivo para el ID
            Buscar módulos por ID en la base de datos
            Mientras no existan módulos con el ID y el ID no sea 0:
                Mostrar un mensaje de error indicando que no existe ningún
módulo con ese ID
                Pedir un nuevo valor para el ID
                Buscar módulos por el nuevo ID en la base de datos
            Devolver el ID

        eliminar(id):
            Intentar:
                Buscar los módulos por ID en la base de datos
                Si no hay módulos con el ID, mostrar un mensaje de
advertencia
                Sino:
                    Abrir una nueva sesión
                    Iniciar una transacción
                    Para cada módulo encontrado:
                        Crear una instancia de gsMatriculas y eliminar las
matrículas asociadas al módulo
                        Eliminar el módulo de la base de datos
                    Confirmar la transacción
                    Mostrar un mensaje de éxito
            Capturar Exception e:
                Mostrar un mensaje de error indicando que no se pudo
eliminar el módulo
            Finalmente:
                Cerrar la sesión

        buscar(campo, valor):
            Abrir una nueva sesión
            Crear una consulta para obtener módulos por el campo y valor
dados
            Establecer el parámetro de la consulta con el valor
proporcionado
            Ejecutar la consulta y devolver la lista de módulos encontrados
            Cerrar la sesión
```

gsMatriculas

```
Clase gsMatriculas extiende gestorDB:
    Atributos:
        - rc (instancia de ReadClient)
        - gsAl (instancia de gsAlumnos)
        - gsModulos (instancia de gsModulos)

    Métodos:
        qualificar():
            Obtener una matrícula mediante obtenerMatriculaPorInput()
            Si la matrícula existe:
                Pedir el número de notas a añadir
                Por cada nota a añadir:
                    Pedir la nota en un rango de 0 a 10
                Añadir las notas a la matrícula
                Llamar a updateNotas para guardar las notas en la base de
datos
                Mostrar un mensaje de éxito
            Sino:
                Mostrar un mensaje de error indicando que el alumno no está
registrado en ese módulo

        modificar():
            Obtener una matrícula mediante obtenerMatriculaPorInput()
            Si la matrícula existe:
                Obtener las notas de la matrícula
                Pedir el identificador de la nota o 0 para mostrar las
notas
                Mientras el identificador sea 0:
                    Mostrar las notas de la matrícula
                    Pedir el identificador de la nota o 0 para mostrar las
notas
                Restar 1 al identificador para adaptarlo a la lista de
notas
                Pedir la nueva nota en un rango de 0 a 10
                Actualizar la nota en la lista de notas
                Llamar a updateNotas para guardar las notas modificadas en
la base de datos
                Mostrar un mensaje de éxito
            Sino:
                Mostrar un mensaje de error indicando que el alumno no está
registrado en ese módulo

        mostrarNotas():
```

```
            Obtener una matrícula mediante obtenerMatriculaPorInput()
            Si la matrícula existe:
                Mostrar las notas de la matrícula
            Sino:
                Mostrar un mensaje de error indicando que el alumno no está
registrado en ese módulo

        mostrarAlumno():
            Pedir el NIA del alumno
            Si el NIA es diferente de 0:
                Buscar al alumno por NIA
                Si el alumno existe:
                    Buscar todas las matrículas del alumno
                    Mostrar las notas de cada matrícula
            Sino:
                Mostrar un mensaje de advertencia indicando que se ha
cancelado la opción

        mostrarAll():
            Abrir una nueva sesión
            Crear una consulta para obtener todas las matrículas
            Obtener la lista de matrículas
            Mostrar las notas de cada matrícula
            Cerrar la sesión

        obtenerMatriculaPorInput():
            Pedir el NIA del alumno y el ID del módulo
            Si el NIA y el ID son diferentes de 0:
                Buscar al alumno por NIA
                Buscar el módulo por ID
                Si el alumno y el módulo existen:
                    Buscar la matrícula para el alumno en el módulo
                    Devolver la matrícula encontrada o null si no existe

        updateNotas(matr):
            Intentar:
                Abrir una nueva sesión
                Iniciar una transacción
                Actualizar la matrícula en la base de datos
                Confirmar la transacción
            Capturar HibernateException e:
                Mostrar un mensaje de error indicando que no se pudieron
actualizar las notas
            Finalmente:
                Cerrar la sesión
```

```
        delMatriculaPorNia(nia):
            Llamar a delMatricula con el campo "alumno.nia" y el valor del
NIA

        delMatriculaIdModulo(id):
        Llamar a delMatricula con el campo "modulo.id" y el valor del ID
del módulo

        buscarModulo(id):
            Llamar a buscar con la clase Modulo y el campo "id" y valor del
ID

        buscarAlumno(nia):
            Llamar a buscar con la clase Alumno y el campo "nia" y valor
del
```

Package app

menus

```
Clase menus:
    Atributos:
        - rc (instancia de ReadClient)
        - gsAl (instancia de gsAlumnos)
        - gsMod (instancia de gsModulos)
        - gsMat (instancia de gsMatriculas)

    Métodos:
        mainMenu():
            Repetir mientras repit sea verdadero:
                Mostrar las opciones del menú principal
                Pedir al usuario que seleccione una opción
                Según la opción seleccionada:
                    Caso 0:
                        Mostrar un mensaje de despedida
                        Establecer repit como falso
                    Caso 1:
                        Llamar a menuAlumnos()
                    Caso 2:
                        Llamar a menuModulos()
                    Caso 3:
                        Llamar a menuMatriculas()
                    Por defecto:
```

```
                Mostrar un mensaje de advertencia

menuAlumnos():
    Inicializar menu como 0
    Repetir mientras repetir sea verdadero:
        Mostrar las opciones del menú de alumnos
        Pedir al usuario que seleccione una opción
        Según la opción seleccionada:
            Caso 0:
                Mostrar un mensaje de despedida
                Establecer repetir como falso
            Caso 1:
                Llamar a gsAl.alta()
            Caso 2:
                Llamar a gsAl.baja()
            Caso 3:
                Llamar a gsAl.mostrarAlumnos()
            Por defecto:
                Mostrar un mensaje de advertencia

menuModulos():
    Inicializar menu como 0
    Repetir mientras repetir sea verdadero:
        Mostrar las opciones del menú de módulos
        Pedir al usuario que seleccione una opción
        Según la opción seleccionada:
            Caso 0:
                Mostrar un mensaje de despedida
                Establecer repetir como falso
            Caso 1:
                Llamar a gsMod.alta()
            Caso 2:
                Llamar a gsMod.baja()
            Caso 3:
                Llamar a gsMod.mostrarModulos()
            Caso 4:
                Llamar a gsMod.matricular()
            Por defecto:
                Mostrar un mensaje de advertencia

menuMatriculas():
    Inicializar menu como 0
    Repetir mientras repetir sea verdadero:
        Mostrar las opciones del menú de matrículas
        Pedir al usuario que seleccione una opción
```

```
                Según la opción seleccionada:
                    Caso 0:
                        Mostrar un mensaje de despedida
                        Establecer repetir como falso
                    Caso 1:
                        Llamar a gsMat.qualificar()
                    Caso 2:
                        Llamar a gsMat.modificar()
                    Caso 3:
                        Llamar a gsMat.mostrarNotas()
                    Caso 4:
                        Llamar a gsMat.mostrarAlumno()
                    Caso 5:
                        Llamar a gsMat.mostrarAll()
                    Por defecto:
                        Mostrar un mensaje de advertencia
```

app

```
Clase app:
    Método main():
        Crear una instancia de la clase menus llamada programa
        Intentar realizar lo siguiente:
            Llamar al método mainMenu() de la instancia programa
        Capturar cualquier excepción (Exception e):
            Mostrar un mensaje de error inesperado
```

Como

Clase Alumno

Esta clase se define como entidad para Hibernate y se asigna a la tabla `Alumnos` de la base de datos. Sus parámetros son: `id`, `nombre`, `apellidos`, `fecha de nacimiento` y `nia`. Estos parámetros se definen como columnas de la tabla para que se guarden los datos en cada una de las columnas respectivas. El `id` está anotado con `@Id` y `@GeneratedValue(strategy = GenerationType.IDENTITY)` para que se genere automáticamente (autoincremental) por la base de datos al agregar un nuevo registro. Dentro de esta clase, tenemos los métodos getters y setters para los parámetros. Es importante destacar que para la fecha, la función pide tres parámetros: el día, el mes y el año.

Clase Modulo

La lógica de esta clase es similar a la de Alumnos, pero cambian los parámetros de la clase. Esta solo tiene un nombre y el `id` que se define por la base de datos.

Clase Matricula

Esta clase `Matricula` está anotada como una entidad de Hibernate y se asigna a la tabla `matriculas` en la base de datos. Similar a las otras clases, tiene atributos como `id`, `alumno`, `modulo` y `notas`. Las anotaciones `@ManyToOne` y `@JoinColumn` se utilizan para establecer las relaciones entre las tablas en la base de datos.

- `@ManyToOne`: se utiliza para indicar la asociación muchos a uno entre dos entidades, como en este caso, que puedes tener muchas matrículas de un alumno o de un módulo.
- `@JoinColumn`: se utiliza para especificar la columna de la tabla propietaria que se utilizará para la asociación. `MAT_ALM_ID` y `MAT_MOD_ID` actúan como claves foráneas a las tablas alumnos y módulos.

La columna `MAT_NOTAS` almacena las notas en formato de cadena, y se proporcionan métodos para trabajar con estas notas, como `getNotasList()` que convierte el string en un array o `setNotasList()` que modifica el string pasándole un array. Además, tengo las funciones `addNota()` que obtiene una lista de notas y les añade las notas que le dan como parámetro, y luego modifica el string de notas, y el método `mostrarNotas()`.

Clase Conexion

Su función principal es crear y configurar la fábrica de sesiones (`SessionFactory`). La fábrica de sesiones es una interfaz en Hibernate que proporciona sesiones para la interacción con la base de datos. La clase `Configuration` crea una nueva configuración obteniendo esa configuración del archivo `hibernate.cfg.xml` y luego crea la fábrica de sesiones. Si hubiera algún problema, captura su excepción. Esta clase tiene un método para obtener la `SessionFactory`, `getSessionFactory()`.

Clase gestorDB

Esta clase solo tiene un método, pero está implementada para hacer simples transacciones con la base de datos que puede que en un futuro ya se implementen. Este método obtiene una instancia de `Session` a través del método `getSessionFactory()`, que en Hibernate se utiliza para realizar operaciones de lectura y escritura en la base de datos. Luego se inicia una transacción que sirve para hacer operaciones y poder guardar los datos en la base de datos. Después se llama al método `save` de la sesión que le doy como parámetro de entrada un objeto para que lo guarde en la base de datos, pero hasta que no acepte la transacción con un `commit`, estos cambios no serán aplicados y finalmente se cierra la sesión.

Clase gsAlumnos

De esta clase explicaré un poco lo que hace cada método, ya que son similares a la práctica anterior pero con conexiones a la base de datos con Hibernate y consultas para obtener los datos.

- **Alta()**: Solicita al usuario información (nombre, apellidos, NIA, fecha de nacimiento) para crear un objeto Alumno, lo añade a la base de datos usando `super.addDB(alumno)`, y muestra un mensaje de éxito.

- **Baja()**: Solicita el NIA del alumno a eliminar, busca al alumno por NIA, elimina cualquier matrícula asociada y al alumno de la base de datos. Maneja posibles errores y muestra mensajes correspondientes.
- **MostrarAlumnos()**: Abre una sesión de Hibernate, ejecuta una consulta con `createQuery` pasándole la consulta y el tipo de resultado de la consulta. Luego concatena con la consulta una función `setParameter` que sirve para indicar a qué hace referencia el string del parámetro que ha introducido y luego indica que solo es un resultado, porque solo puede haber un alumno con ese NIA. Imprime cada los datos de ese alumno y cierra la sesión.
- **PedirNia(boolean exist)**: Solicita al usuario el NIA del alumno, permitiendo cancelar y verifica si el NIA ya existe.
- **BuscarAlmNia(int nia)**: Abre una sesión de Hibernate, realiza una consulta para obtener el alumno con el NIA dado y devuelve el resultado.
- **ComprobarNia(int nia, boolean exist)**: Abre una sesión de Hibernate, ejecuta una consulta para obtener todos los NIA existentes y verifica si el NIA dado ya existe en la base de datos (si exist es true) o no (si exist es false).

Clase gsModulos

De la misma manera que antes, explicaré lo que hacen las funciones de esta clase y detallaré alguna función que no he explicado aún.

- **Alta()**: Solicita al usuario el nombre de un nuevo módulo, crea un objeto Modulo con ese nombre, lo añade a la base de datos usando `super.addDB(m)`, y muestra un mensaje de éxito.
- **Baja()**: Solicita al usuario el ID del módulo a eliminar, busca y elimina todos los módulos con ese ID de la base de datos, así como cualquier matrícula asociada. Muestra mensajes de éxito o cancelación.
- **Matricular()**: Solicita al usuario el NIA del alumno a matricular y el ID del módulo al que desea matricularlo. Busca al alumno y el módulo correspondientes, verifica si ya está matriculado, y si no, crea una nueva matrícula y la añade a la base de datos. Muestra mensajes de éxito o error.
- **MostrarModulos()**: Abre una sesión de Hibernate, ejecuta una consulta para obtener todos los módulos de la base de datos, imprime cada módulo y cierra la sesión.
- **PedirId()**: Solicita al usuario el ID del módulo, permitiendo cancelar. Verifica si el ID existe en la base de datos y devuelve el ID válido.
- **Eliminar(int id)**: Elimina todos los módulos con el ID dado, así como las matrículas asociadas a esos módulos. Muestra mensajes de éxito o error.
- **Buscar(String campo, int valor)**: Abre una sesión de Hibernate, ejecuta una consulta para obtener los módulos que cumplen con el valor del campo proporcionado y devuelve la lista de resultados. Cierra la sesión después de obtener los resultados. Esta es la función que suelo usar en esta clase para hacer consultas a la base de datos.

Clase gsMatriculas

Al igual que antes, explicaré las funciones.

- **qualificar():** Solicita al usuario el NIA del alumno y el ID del módulo para obtener la matrícula correspondiente. Luego, pide al usuario ingresar las notas para esa matrícula y las actualiza en la base de datos.
- **modificar():** Solicita al usuario el NIA del alumno y el ID del módulo para obtener la matrícula correspondiente. Luego, permite al usuario seleccionar una nota específica para modificar y actualiza esa nota en la base de datos.
- **mostrarNotas():** Solicita al usuario el NIA del alumno y el ID del módulo para obtener la matrícula correspondiente y mostrar todas las notas asociadas a esa matrícula.
- **mostrarAlumno():** Solicita al usuario el NIA del alumno y muestra todas las notas asociadas a ese alumno en cualquier módulo.
- **mostrarAll():** Muestra todas las notas de todos los alumnos en todos los módulos.
- **obtenerMatriculaPorInput():** Solicita al usuario el NIA del alumno y el ID del módulo para obtener la matrícula correspondiente.
- **updateNotas(Matricula matr):** Actualiza las notas de una matrícula en la base de datos.
- **delMatriculaPorNia(int nia):** Elimina todas las matrículas asociadas a un alumno por su NIA.
- **delMatriculaIdModulo(int id):** Elimina todas las matrículas asociadas a un módulo por su ID.
- **buscarModulo(int id):** Busca un módulo por su ID.
- **buscarAlumno(int nia):** Busca un alumno por su NIA.
- **buscarMatricula(Alumno alm, Modulo mod):** Busca una matrícula por un alumno y un módulo específicos.
- **buscar(Class entidad, String campo, Object valor):** Realiza una búsqueda general en la base de datos de la entidad proporcionada según el campo y el valor proporcionados. Este método utiliza programación genérica (`<T>`) para poder trabajar con diferentes clases.
- **delMatricula(String campo, Object valor, String descripcion):** Elimina matrículas en función de un campo y valor específicos, mostrando mensajes de advertencia o éxito.

Clase menus

Estas funciones organizan las opciones disponibles para el usuario en distintos menús y gestionan la ejecución de acciones relacionadas con alumnos, módulos y evaluaciones, conectándose con las clases `gsAlumnos`, `gsModulos` y `gsMatriculas` para realizar las operaciones correspondientes en la base de datos.

- **mainMenu():** Esta función representa el menú principal del programa. Utiliza un bucle para mostrar las opciones disponibles. El usuario elige una opción y se ejecuta la acción correspondiente. Si selecciona "0", el programa se cierra.

- **menuAlumnos():** Aquí, se presenta un submenú dedicado a las operaciones con alumnos: alta, baja y listar. El usuario selecciona una opción. Si elige "0", sale del menú de alumnos.
- **menuModulos():** Similar al anterior, este método muestra un submenú para operaciones con módulos: alta, baja, listar y matricular alumno. El usuario selecciona una opción. Si elige "0", sale del menú de módulos.
- **menuMatriculas():** Este método presenta un submenú para las operaciones de evaluación: calificar, modificar, mostrar notas de un módulo, mostrar notas de un alumno y mostrar todas las notas del centro. El usuario selecciona una opción. Si elige "0", sale del menú de evaluación.

Clase app

Esta clase Java app contiene el método principal `main`, que instancia la clase `menus` y llama a su método `mainMenu` para iniciar la ejecución del programa de gestión.

Conclusión

En conclusión, considero que esta actividad ha sido interesante y ha contribuido significativamente a mejorar mis habilidades de programación con Hibernate y bases de datos. El uso de nuevas funciones para acceder y modificar la base de datos desde mi programa ha ampliado mi comprensión y destrezas en este ámbito. A lo largo de la actividad, he descubierto algunas funciones que no conocía; mientras que algunas las he implementado con éxito, otras no han sido tan accesibles debido a limitaciones en mi comprensión.
La práctica de las operaciones CRUD con Hibernate ha sido beneficiosa y ha reforzado mi conocimiento en el manejo de bases de datos. Además, me intriga conocer las diferentes aproximaciones que mis compañeros han tomado para abordar esta actividad, ya que estoy consciente de que hay varias formas de implementarla. Consultar a mis amigos y obtener explicaciones sobre sus enfoques podría proporcionarme valiosas perspectivas y aprender nuevas técnicas.

En general, aunque la actividad fue fácil de entender conceptualmente, la implementación resultó ser desafiante debido a su extensión y la posibilidad de errores. Sin embargo, estoy satisfecho con los resultados obtenidos y considero que la dificultad fue proporcional al aprendizaje adquirido.