UNIVERSIDAD SIGLO 21

INF275-10156 - SEMINARIO DE PRÁCTICA DE INFORMATICA

Trabajo Práctico N°4

Rolando Andrés Palermo

DNI: 33.311.173 / Legajo: VINF013533

Profesor: **PABLO ALEJANDRO VIRGOLINI**

30 de Junio de 2024

01 OBJETIVO

Esta actividad busca que seas capaz de plantear una solución problemática que pueda resolverse mediante la realización de un proyecto informático.

Para llevar adelante este desafío, vas a poder aplicar muchos de los conceptos más importantes abordados durante el desarrollo del módulo 4, que recupera tu trabajo en materias troncales de la carrera.

Durante el proceso, lograrás aplicar tus conocimientos para alcanzar los siguientes objetivos:

- Desarrollar un proyecto integrador para solucionar un problema.
- Identificar el patrón de diseño adecuado según las características de un problema.
- Utilizar arreglos y ArrayList.
- Realizar la persistencia de datos en una base de datos MySQL.
- Emplear archivos para guardar y recuperar información relevante.
- Realizar un video para comunicar el alcance de tu proyecto.

En este cuarto trabajo, deberás retomar el proyecto que estás desarrollando y avanzar con la versión final integradora.

02 SITUACIÓN PROBLEMÁTICA

Lee con atención el siguiente caso teniendo presente cada uno de los contenidos que hemos desarrollado en el módulo. Una vez leído, tendrás que resolver preguntas cerradas en base al mismo. Haz clic sobre el siguiente enlace para descargar el enunciado:

Click aquí

03 CONSIGNA

Considera que continúas trabajando en la organización cuyo problema definiste resolver con un proyecto informático, y estás en el equipo de desarrollo que va a llevar adelante el mismo.

Recuerda que el proyecto informático definido debe ser distinto al considerado en la lectura y que el entregable final debe cumplir con los siguientes objetivos:

- Realizar el análisis del modelo de negocios para definir y justificar el proyecto.
- Aplicar el proceso unificado de desarrollo (PUD) para garantizar la calidad, escalabilidad y eficiencia en el ciclo de desarrollo.
- Utilizar una base de datos MySQL para la persistencia de los datos.
- Emplear Java como lenguaje de programación para el desarrollo del sistema.

A los fines del trabajo, para la implementación de la base de datos y el desarrollo con Java, puedes presentar un prototipo, que es "es la creación de un modelo operacional que incluya solo algunas características del sistema final" (Kendall & Kendall, 2011).

El concepto de operacional es clave, ya que no se trata de un simple modelo, sino que permite desarrollar módulos que se van integrando en la versión final del sistema.

Esta cuarta actividad te permitirá avanzar con el proyecto integrador que debe utilizar un patrón de diseño coherente con las definiciones establecidas en la arquitectura definida.

Para el desarrollo del prototipo en Java, deberás crear un paquete cuyas clases permitan realizar la persistencia en una base de datos MySQL. Esta actividad incluye establecer conexiones, realizar consultas, actualizar registros y presentar resultados en la interfaz. En todos los casos que se requiera, recuerda aplicar correctamente el manejo de excepciones.

El desarrollo debe incluir la aplicación de clases abstractas o interfaces. También debe contemplar la utilización de arreglos y emplear la clase ArrayList, ambas estructuras de forma complementaria.

Según las características del proyecto, puedes utilizar archivos. Aunque esta actividad es opcional, te recomiendo practicar su aplicación, previo a presentarte en el EFIP I.

Los entregables son los siguientes:

- Presentación del desarrollo en Java.
- Entrega del proyecto integrador, completo.
- Presentación del proyecto en un video con una duración de aproximadamente 3 minutos.

El proyecto o prototipo desarrollado en Java debe contemplar necesariamente las siguientes características:

Corrección de las observaciones realizadas en la entrega de las actividades anteriores.
Selección del patrón de diseño y justificación.

- Persistencia y consulta de datos en una base de datos MySQL. Esta actividad incluye establecer conexiones, realizar consultas, actualizar registros y presentar resultados en la interfaz.
- Correcta aplicación de excepciones para la interacción con la base de datos MySQL.
- Inclusión pertinente de clases abstractas o interfaces.
- Utilización complementaria de arreglos y de la clase ArrayList.

En esta instancia, te sugiero revisar la rúbrica para asegurar el cumplimiento del criterio expuesto que considera la evaluación de tu participación activa en actividades propuestas: foros, encuentros sincrónicos, aportes en espacios comunes y comunicación.

Te recomiendo, además, apoyarte en las lecturas y bibliografía propuesta en las materias: Programación Orientada a Objetos, Taller de Algoritmos y Estructuras de Datos I.

Referencia:

Kendall, K., & Kendall, J. (2011). Análisis y diseño de sistemas. Pearson Education.

04 FORMATO ENTREGABLE

La actividad propuesta debe ser elaborada en forma individual y respetar los criterios de presentación mencionados a continuación. Se debe realizar una entrega incremental, considerando las correcciones realizadas en las entregas previas.

El informe se debe presentar en un documento en formato PDF (papel A4, letra Calibri 11, espaciado simple). Debe incluir las respuestas a las consignas de forma prolija y ordenada, con su correspondiente explicación para justificar las decisiones adoptadas. Incluir una portada y respetar el siguiente criterio para el nombre del archivo:

«APELLIDO-NOMBRE»-AP4.PDF

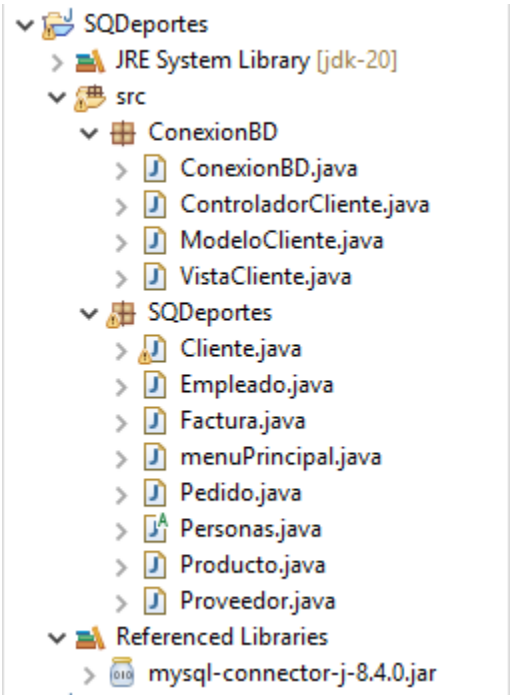
La realización de todos los diagramas requiere utilizar una herramienta de libre elección, pero asegurando la prolijidad y claridad de parentación. Todos los diagramas deben incluirse en el documento a entregar con su correspondiente explicación.

Para la persistencia de los datos, deberás utilizar una base de datos MySQL, y para el desarrollo del prototipo, una IDE de Java. La explicación debe estar en el documento a entregar e incluir un enlace a la versión completa (código y archivos asociados) en Github: https://github.com/Links to an external site..

Se evaluará la precisión en las respuestas, capacidad de síntesis, redacción, ortografía y calidad de la presentación.

En caso de requerirlo, es importante revisar el cumplimiento de las normas de estilo American Psychological Association (APA).

03 RESOLUCIÓN



Aquí se aplicaron los conceptos de conexión a bases de datos y patrones MVC (Modelo-Vista-Controlador) al proyecto SQDeportes donde se creó un package llamado ConexionBD y las siguientes clases:

Clase “ConexionBD” para manejar la conexión a la base de datos.
Clase “ModeloCliente” que maneje la lógica de acceso a datos para los clientes.
Clase “VistaCliente” que maneje la interacción con el usuario.
Clase “ControladorCliente” que maneje la lógica de negocio.

Modificación del Menú Principal:
Se modificó la clase menuPrincipal para integrar el controlador de clientes con la base de datos, utilizando los métodos de las clases anterior mencionadas.

1- **Persistencia y consulta de datos en una base de datos MySQL:**

- La clase ConexionBD establece la conexión con la base de datos MySQL usando JDBC.

```java
package ConexionBD;

import java.sql.Connection;

public class ConexionBD {
    // Constantes para la URL de conexión, usuario y contraseña de la base de datos
    private static final String URL = "jdbc:mysql://localhost:3306/sqdeportes";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    // Método estático para establecer una conexión a la base de datos
    public static Connection conectar() {
        Connection conexion = null; // Variable para almacenar la conexión
        try {
            // Intentar establecer una conexión utilizando los parámetros proporcionados
            conexion = DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (SQLException e) {
            // Capturar y mostrar cualquier excepción de SQL que ocurra durante la conexión
            System.out.println("Error al conectar a la base de datos: " + e.getMessage());
        }
        return conexion; // Devolver la conexión (puede ser null si ocurrió un error)
    }
}
```

- En ModeloCliente, se implementan métodos para insertar, eliminar, modificar, consultar y listar clientes en la base de datos.

```java
    // Método para insertar un nuevo cliente en la base de datos
    public void insertarCliente(Cliente cliente) throws SQLException {
        String sql = "INSERT INTO clientes (nombre, DNI, direccion, telefono, localidad, provincia, cuentaCorriente) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement stmt = conexion.prepareStatement(sql)) {
            stmt.setString(1, cliente.getNombre());
            stmt.setString(2, cliente.getDNI());
            stmt.setString(3, cliente.getDireccion());
            stmt.setString(4, cliente.getTelefono());
            stmt.setString(5, cliente.getLocalidad());
            stmt.setString(6, cliente.getProvincia());
            stmt.setDouble(7, cliente.getCuentaCorriente());
            stmt.executeUpdate(); // Ejecuta la inserción
        }
    }

    // Método para eliminar un cliente de la base de datos utilizando su ID
    public void eliminarCliente(int idCliente) throws SQLException {
        String sql = "DELETE FROM clientes WHERE idCliente = ?";
        try (PreparedStatement stmt = conexion.prepareStatement(sql)) {
            stmt.setInt(1, idCliente);
            stmt.executeUpdate(); // Ejecuta la eliminación
        }
    }

    // Método para modificar los datos de un cliente existente en la base de datos
    public void modificarCliente(Cliente cliente) throws SQLException {
        String sql = "UPDATE clientes SET nombre = ?, DNI = ?, direccion = ?, telefono = ?, localidad = ?, provincia = ?, cuentaCorriente = ? WHERE idCliente = ?";
        try (PreparedStatement stmt = conexion.prepareStatement(sql)) {
            stmt.setString(1, cliente.getNombre());
            stmt.setString(2, cliente.getDNI());
            stmt.setString(3, cliente.getDireccion());
            stmt.setString(4, cliente.getTelefono());
            stmt.setString(5, cliente.getLocalidad());
            stmt.setString(6, cliente.getProvincia());
            stmt.setDouble(7, cliente.getCuentaCorriente());
            stmt.setInt(8, cliente.getIdCliente());
            stmt.executeUpdate(); // Ejecuta la actualización
        }
    }
```

2- Correcta aplicación de excepciones para la interacción con la base de datos MySQL:

Se utilizan excepciones (SQLException) en los métodos que interactúan directamente con la base de datos, como insertarCliente, eliminarCliente, modificarCliente, consultarCliente, y listarClientes.

insertarCliente, eliminarCliente y modificarCliente ya se encuentran en la imagen anterior.

```java
// Método para consultar los datos de un cliente en la base de datos utilizando su ID
public Cliente consultarCliente(int idCliente) throws SQLException {
    String sql = "SELECT * FROM clientes WHERE idCliente = ?";
    try (PreparedStatement stmt = conexion.prepareStatement(sql)) {
        stmt.setInt(1, idCliente);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            // Si se encuentra el cliente, crear y devolver un objeto Cliente con sus datos
            return new Cliente(
                rs.getInt("idCliente"),
                rs.getString("nombre"),
                rs.getString("direccion"),
                rs.getString("telefono"),
                rs.getString("localidad"),
                rs.getString("provincia"),
                rs.getInt("idCliente"),
                rs.getString("DNI"),
                rs.getDouble("cuentaCorriente")
            );
        }
    }
    return null; // Si no se encuentra el cliente, devolver null
}

// Método para listar todos los clientes de la base de datos
public List<Cliente> listarClientes() throws SQLException {
    List<Cliente> clientes = new ArrayList<>();
    String sql = "SELECT * FROM clientes";
    try (PreparedStatement stmt = conexion.prepareStatement(sql);
         ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            // Agregar cada cliente encontrado a la lista de clientes
            clientes.add(new Cliente(
                rs.getInt("idCliente"),
                rs.getString("nombre"),
                rs.getString("direccion"),
                rs.getString("telefono"),
                rs.getString("localidad"),
                rs.getString("provincia"),
                rs.getInt("idCliente"),
                rs.getString("DNI"),
```

3- Inclusión pertinente de clases abstractas o interfaces:

La clase abstracta Persona, que es heredada por la clase Cliente, podemos ver que se cumple el punto sobre la inclusión de clases abstractas o interfaces.

4- Utilización complementaria de arreglos y de la clase ArrayList:

Se utiliza la clase ArrayList en ModeloCliente para almacenar y devolver listas de clientes en los métodos listarClientes.
Aquí podemos ver que se emplean listas (ArrayList) para manejar colecciones de datos.

```java
    // Método para listar todos los clientes de la base de datos
    public List<Cliente> listarClientes() throws SQLException {
        List<Cliente> clientes = new ArrayList<>();
        String sql = "SELECT * FROM clientes";
        try (PreparedStatement stmt = conexion.prepareStatement(sql);
             ResultSet rs = stmt.executeQuery()) {
            while (rs.next()) {
                // Agregar cada cliente encontrado a la lista de clientes
                clientes.add(new Cliente(
                    rs.getInt("idCliente"),
                    rs.getString("nombre"),
                    rs.getString("direccion"),
                    rs.getString("telefono"),
                    rs.getString("localidad"),
                    rs.getString("provincia"),
                    rs.getInt("idCliente"),
                    rs.getString("DNI"),
                    rs.getDouble("cuentaCorriente")
                ));
            }
        }
        return clientes; // Devolver la lista de clientes
    }
}
```

Conclusión.
El código muestra una implementación funcional de persistencia y consulta de datos en una base de datos MySQL, manejo adecuado de excepciones para operaciones de base de datos, y utiliza colecciones como ArrayList para almacenamiento de datos. Aunque no utiliza clases abstractas o interfaces directamente en este código específico, sigue buenas prácticas de diseño modular y separación de responsabilidades entre la vista (VistaCliente), el controlador (ControladorCliente), y el modelo (ModeloCliente).

Link de github donde se subirá un .zip con todo el proyecto de SQDeportes:
https://github.com/rolandoandres22/tp4-seminario-de-practica.git