MANUAL TÉCNICO

ADMINISTRADOR DE EVENTO TRIFORCE SOFTWARE
PRACTICA 1 IPC2

APLICACIÓN DE ESCRITORIO

CRISTIAN ALEJANDRO ROLDÁN LÓPEZ
202147280

Especificaciones

Versión de Java utilizado: Java 21.
Versión de JDK: 21
Sistema Operativo Utilizado: Ubuntu 24.04 LTS
IDE utilizado: Apache NetBeans IDE 21.
Gestor de base de datos: Mysql 8.0.43

Métodos utilizados

Se utilizo una lógica en la que se crearon entidades por cada tabla de la base de datos y controladores para manejar la lógica en la que se realizarían las consultas también el apartado de formularios que serian las vistas.

Clase EntidadActividad

```
public class EntidadActividad {

    private String codigoActividad;
    private String codigoEvento;
    private TipoCharla tipoCharla;
    private String titulo;
    private String idParticipante;
    private LocalTime horaInicio;
    private LocalTime horaFin;
    private int cupoMaximo;

    public EntidadActividad(String codigoActividad, String codigoEvento, TipoCharla
tipoCharla, String titulo, String idParticipante, LocalTime horaInicio, LocalTime horaFin, int
cupoMaximo) {
        this.codigoActividad = codigoActividad;
        this.codigoEvento = codigoEvento;
        this.tipoCharla = tipoCharla;
        this.titulo = titulo;
        this.idParticipante = idParticipante;
        this.horaInicio = horaInicio;
        this.horaFin = horaFin;
        this.cupoMaximo = cupoMaximo;
    }

    public String getCodigoActividad() {
        return codigoActividad;
    }

    public void setCodigoActividad(String codigoActividad) {
        this.codigoActividad = codigoActividad;
    }

    public String getCodigoEvento() {
        return codigoEvento;
    }

    public void setCodigoEvento(String codigoEvento) {
        this.codigoEvento = codigoEvento;
    }
```

```
public TipoCharla getTipoCharla() {
    return tipoCharla;
}

public void setTipoCharla(TipoCharla tipoCharla) {
    this.tipoCharla = tipoCharla;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getIdParticipante() {
    return idParticipante;
}

public void setIdParticipante(String idParticipante) {
    this.idParticipante = idParticipante;
}

public LocalTime getHoraInicio() {
    return horaInicio;
}

public void setHoraInicio(LocalTime horaInicio) {
    this.horaInicio = horaInicio;
}

public LocalTime getHoraFin() {
    return horaFin;
}

public void setHoraFin(LocalTime horaFin) {
    this.horaFin = horaFin;
}

public int getCupoMaximo() {
    return cupoMaximo;
}

public void setCupoMaximo(int cupoMaximo) {
    this.cupoMaximo = cupoMaximo;
}
```

```
}
```

de esta manera se trabajaron las demás entidades

```
public class EntidadAsistencia {

    private String idParticipante;
    private String codigoActividad;
    private String correoParticipante;

    public EntidadAsistencia(String correoParticipante, String codigoActividad) {
        this.correoParticipante = correoParticipante;
        this.codigoActividad = codigoActividad;
    }

    public String getIdParticipante() {
        return idParticipante;
    }

    public void setIdParticipante(String idParticipante) {
        this.idParticipante = idParticipante;
    }

    public String getCodigoActividad() {
        return codigoActividad;
    }

    public void setCodigoActividad(String codigoActividad) {
        this.codigoActividad = codigoActividad;
    }

    public String getCorreoParticipante() {
        return correoParticipante;
    }

    public void setCorreoParticipante(String correoParticipante) {
        this.correoParticipante = correoParticipante;
    }



}

public class EntidadCertificado {

    private String correoParticipante;
    private String codigoEvento;
    private String nombreParticipante;
    private String tituloEvento;
```

```
private String fechaEvento;
private String codigoActividad;

public EntidadCertificado(String correoParticipante, String codigoActividad) {
    this.correoParticipante = correoParticipante;
    this.codigoActividad = codigoActividad;
}

public String getCorreoParticipante() {
    return correoParticipante;
}

public void setCorreoParticipante(String correoParticipante) {
    this.correoParticipante = correoParticipante;
}

public String getCodigoEvento() {
    return codigoEvento;
}

public void setCodigoEvento(String codigoEvento) {
    this.codigoEvento = codigoEvento;
}

public String getNombreParticipante() {
    return nombreParticipante;
}

public void setNombreParticipante(String nombreParticipante) {
    this.nombreParticipante = nombreParticipante;
}

public String getTituloEvento() {
    return tituloEvento;
}

public void setTituloEvento(String tituloEvento) {
    this.tituloEvento = tituloEvento;
}

public String getFechaEvento() {
    return fechaEvento;
}

public void setFechaEvento(String fechaEvento) {
    this.fechaEvento = fechaEvento;
}

public String getCodigoActividad() {
```

```
        return codigoActividad;
    }

    public void setCodigoActividad(String codigoActividad) {
        this.codigoActividad = codigoActividad;
    }









}

public class EntidadEvento {

    private String codigo;
    private LocalDate fechaEvento;
    private TipoEvento tipoEvento;
    private String titulo;
    private String ubicacion;
    private int cupoMaximo;
    private double pagoEVento;

    public EntidadEvento(String codigo, LocalDate fechaEvento, TipoEvento tipoEvento, String
titulo, String ubicacion, int cupoMaximo, double pagoEVento) {
        this.codigo = codigo;
        this.fechaEvento = fechaEvento;
        this.tipoEvento = tipoEvento;
        this.titulo = titulo;
        this.ubicacion = ubicacion;
        this.cupoMaximo = cupoMaximo;
        this.pagoEVento = pagoEVento;
    }

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public LocalDate getFechaEvento() {
        return fechaEvento;
    }

    public void setFechaEvento(LocalDate fechaEvento) {
        this.fechaEvento = fechaEvento;
```

```
    }

    public TipoEvento getTipoEvento() {
        return tipoEvento;
    }

    public void setTipoEvento(TipoEvento tipoEvento) {
        this.tipoEvento = tipoEvento;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getUbicacion() {
        return ubicacion;
    }

    public void setUbicacion(String ubicacion) {
        this.ubicacion = ubicacion;
    }

    public int getCupoMaximo() {
        return cupoMaximo;
    }

    public void setCupoMaximo(int cupoMaximo) {
        this.cupoMaximo = cupoMaximo;
    }

    public double getPagoEVento() {
        return pagoEVento;
    }

    public void setPagoEVento(double pagoEVento) {
        this.pagoEVento = pagoEVento;
    }


}
```

Controlador de actividades:

```
public class RegistrarActividad extends ConectarDBA {

    public RegistrarActividad() {
        super();
        connect();
    }

    public boolean agregarActividad(EntidadActividad entidadActividad) {

        int idParticipante = -1;
        String consultaParticipante = "SELECT p.idParticipante "
                + "FROM registro_participante p "
                + "INNER JOIN inscripcion i ON p.idParticipante = i.idParticipante "
                + "WHERE p.Correo = ? AND i.tipoInscripcion <> 'ASISTENTE' AND i.codigoEvento
= ?";

        try (PreparedStatement ps = getConnect().prepareStatement(consultaParticipante)) {
            ps.setString(1, entidadActividad.getIdParticipante());
            ps.setString(2, entidadActividad.getCodigoEvento());
            ResultSet resultado = ps.executeQuery();
            if (resultado.next()) {
                idParticipante = resultado.getInt("idParticipante");
            } else {
                System.out.println("Participante no válido o solo ASISTENTE: " +
entidadActividad.getIdParticipante());
                return false;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }

        String insertarActividad = "INSERT INTO registrar_actividad "
                + "(codigoActividad, codigoEvento, tipoActividad, tituloActividad, idParticipante,
horaInicio, horaFin, cupoMaximo) "
                + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

        try (PreparedStatement psInsert = getConnect().prepareStatement(insertarActividad)) {
            psInsert.setString(1, entidadActividad.getCodigoActividad());
            psInsert.setString(2, entidadActividad.getCodigoEvento());
            psInsert.setString(3, entidadActividad.getTipoCharla().name());
            psInsert.setString(4, entidadActividad.getTitulo());
            psInsert.setInt(5, idParticipante);
            psInsert.setTime(6, Time.valueOf(entidadActividad.getHoraInicio()));
            psInsert.setTime(7, Time.valueOf(entidadActividad.getHoraFin()));
            psInsert.setInt(8, entidadActividad.getCupoMaximo());
```

```
            int filas = psInsert.executeUpdate();
            return filas > 0;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

Clase para leer el archvo de texto y agregarlo en la base de datos, recibe como paremtros, la ruta, el tiempo en milisegundos y un Jtextfield para verificar la lectura

```
public class LeerArchivo {

    private final RegistrarEvento eventos = new RegistrarEvento();
    private final RegistroParticipante participantes = new RegistroParticipante();
    private final RegistroInscripcion inscripciones = new RegistroInscripcion();
    private final RegistroPago pagos = new RegistroPago();
    private final RegistrarActividad actividad = new RegistrarActividad();
    private final RegistrarAsistencia asistencia = new RegistrarAsistencia();
    private final Certificado certificados = new Certificado();
    private final ReporteHTML reporteHTML = new ReporteHTML();

    public void leerArchivo(String ruta, int tiempo, JTextField textField) {
        Thread hilo = new Thread(() -> {
            try (BufferedReader br = new BufferedReader(new FileReader(ruta))) {
                String linea;
                while ((linea = br.readLine()) != null) {
                    linea = linea.trim();
                    if (!linea.isEmpty()) {
                        procesarLinea(linea);

                        final String lineaActual = linea;
                        javax.swing.SwingUtilities.invokeLater(()
                                -> textField.setText(textField.getText() + "Procesando: \n" + lineaActual)
                        );

                        Thread.sleep(tiempo);
                    }
                }
            } catch (IOException e) {
                System.out.println("Error leyendo el archivo: " + e.getMessage());
            } catch (InterruptedException e) {
                System.out.println("Ejecución interrumpida");
                Thread.currentThread().interrupt();
            }
```

```
    });
    hilo.start();
}

private void procesarLinea(String linea) {
    if (linea.isEmpty() || linea.startsWith("/*") || linea.startsWith("//")) {
        return;
    }

    int indice = linea.indexOf("(");
    if (indice == -1) {
        return;
    }

    if (linea.endsWith(";")) {
        linea = linea.substring(0, linea.length() - 1);
    }

    String accion = linea.substring(0, indice).trim();
    String contenido = linea.substring(indice + 1, linea.lastIndexOf(")"));
    List<String> datos = separarValores(contenido);

    switch (accion) {
        case "REGISTRO_EVENTO":
            DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy");
            EntidadEvento evento = new EntidadEvento(
                    datos.get(0),
                    LocalDate.parse(datos.get(1), formato),
                    TipoEvento.valueOf(datos.get(2).toUpperCase()),
                    datos.get(3),
                    datos.get(4),
                    Integer.parseInt(datos.get(5)),
                    Double.parseDouble(datos.get(6))
            );
            eventos.agregarEvento(evento);
            System.out.println("Evento registrado: " + datos);
            break;

        case "REGISTRO_PARTICIPANTE":
            EntidadParticipante participante = new EntidadParticipante(
                    datos.get(0),
                    TipoParticipante.valueOf(datos.get(1).toUpperCase()),
                    datos.get(2),
                    datos.get(3)
            );
            participantes.agregarParticipante(participante);
            System.out.println("Participante registrado: " + datos);
            break;
```

```
        case "INSCRIPCION":
            EntidadInscripcion inscripcion = new EntidadInscripcion(
                    datos.get(0),
                    datos.get(1),
                    TipoInscripcion.valueOf(datos.get(2).toUpperCase())
            );
            inscripciones.agregarInscripcion(inscripcion);
            System.out.println("Inscripción registrada: " + datos);
            break;

        case "PAGO":
            EntidadPago pago = new EntidadPago(
                    datos.get(0),
                    datos.get(1),
                    TipoPago.valueOf(datos.get(2).toUpperCase()),
                    Double.parseDouble(datos.get(3))
            );
            pagos.registrarPago(pago);
            System.out.println("Pago registrado: " + datos);
            break;

        case "VALIDAR_INSCRIPCION":

            System.out.println("Validar inscripción: " + datos);
            break;

        case "REGISTRO_ACTIVIDAD":
            DateTimeFormatter horaFormato = DateTimeFormatter.ofPattern("HH:mm");

            EntidadActividad entidadActividad = new EntidadActividad(
                    datos.get(0),
                    datos.get(1),
                    TipoCharla.valueOf(datos.get(2)),
                    datos.get(3),
                    datos.get(4),
                    LocalTime.parse(datos.get(5), horaFormato),
                    LocalTime.parse(datos.get(6), horaFormato),
                    Integer.parseInt(datos.get(7))
            );
            actividad.agregarActividad(entidadActividad);
            System.out.println("Actividad: " + datos);
            break;

        case "ASISTENCIA":
            EntidadAsistencia entidadAsistencia = new EntidadAsistencia(datos.get(0),
datos.get(1));
            asistencia.registrarAsistencia(entidadAsistencia);
            System.out.println("Asistencia: " + datos);
            break;
```

```
        case "CERTIFICADO":
            EntidadCertificado certificado = new EntidadCertificado(datos.get(0), datos.get(1));
            certificados.generarCertificado(certificado);
            System.out.println("Certificado generado: " + datos);
            break;

        case "REPORTE_PARTICIPANTES":
            String codigoEvPart = datos.size() > 0 ? datos.get(0) : "";
            String tipoPart = datos.size() > 1 ? datos.get(1) : "";
            String institucion = datos.size() > 2 ? datos.get(2) : "";

            ControladorGenerarReporte reportePart = new ControladorGenerarReporte();
            List<ReporteParticipante> listaPart =
reportePart.obtenerReporteParticipantes(codigoEvPart, tipoPart, institucion);

            System.out.println("Reporte Participantes:");
            for (ReporteParticipante rp : listaPart) {
                System.out.println(rp.getNombreCompleto() + " - " + rp.getCorreo() + " - " +
rp.getTipo());
            }
            break;

        case "REPORTE_ACTIVIDADES":
            String codigoEvAct = datos.size() > 0 ? datos.get(0) : "";
            String tipoAct = datos.size() > 1 ? datos.get(1) : "";
            String correoEncarg = datos.size() > 2 ? datos.get(2) : "";

            ControladorGenerarReporte reporteAct = new ControladorGenerarReporte();
            List<ReporteActividad> listaAct =
reporteAct.obtenerActividadesParaReporte(codigoEvAct, tipoAct, correoEncarg);

            System.out.println("Reporte Actividades:");
            for (ReporteActividad ra : listaAct) {
                System.out.println(ra.getTitulo() + " - Encargado: " + ra.getEncargado() + " -
Participantes: " + ra.getCantidadParticipantes());
            }
            break;

        case "REPORTE_EVENTOS":
            ControladorGenerarReporte reporteEvt = new ControladorGenerarReporte();
            List<ReporteEvento> listaEvt = reporteEvt.obtenerReporteEventos();

            System.out.println("Reporte Eventos:");
            for (ReporteEvento re : listaEvt) {
                System.out.println(re.getCodigoEvento() + " - " + re.getNombreParticipante() + " -
Pago: " + re.getMetodoPago());
            }
            break;
```

```
            default:
                System.out.println("Acción desconocida: " + accion);
        }
    }

    private List<String> separarValores(String entrada) {
        List<String> lista = new ArrayList<>();
        StringBuilder actual = new StringBuilder();
        boolean dentroComillas = false;

        for (char c : entrada.toCharArray()) {
            if (c == '"') {
                dentroComillas = !dentroComillas;
            } else if (c == ',' && !dentroComillas) {
                lista.add(actual.toString().trim().replaceAll("^\"|\"$", ""));
                actual.setLength(0);
                continue;
            }
            actual.append(c);
        }
        if (actual.length() > 0) {
            lista.add(actual.toString().trim().replaceAll("^\"|\"$", ""));
        }
        return lista;
    }
}

public class RegistrarEvento extends ConectarDBA {

    private DateTimeFormatter formatoFecha;

    public RegistrarEvento() {
        super();
        connect();
        this.formatoFecha = DateTimeFormatter.ofPattern("dd/MM/yyyy");
    }

    public boolean agregarEvento(EntidadEvento entidadEvento) {

        String queryEvento = "INSERT INTO registro_evento (Codigo, Fecha_Evento,
Tipo_Evento, Titulo, Ubicacion, Cupo_Maximo, Pago_Evento)"
                + "VALUES (?,?,?,?,?,?,?)";

        try (PreparedStatement pstm = getConnect().prepareStatement(queryEvento)) {

            pstm.setString(1, entidadEvento.getCodigo());
            pstm.setString(2, entidadEvento.getFechaEvento().toString());
            pstm.setString(3, entidadEvento.getTipoEvento().name());
```

```
            pstm.setString(4, entidadEvento.getTitulo());
            pstm.setString(5, entidadEvento.getUbicacion());
            pstm.setInt(6, entidadEvento.getCupoMaximo());
            pstm.setDouble(7, entidadEvento.getPagoEVento());

            int filas = pstm.executeUpdate();
            return filas > 0;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public LocalDate fechaValida(String fecha){
        try {
            return LocalDate.parse(fecha, formatoFecha);
        } catch (DateTimeException e) {
            return null;
        }
    }

}
```

Utilzación de clases enum para manejar mejor la lógica

```
public enum TipoEvento {

    CHARLA, TALLER, DEBATE, OTRO;
}

public enum TipoCharla {
    CHARLA, TALLER, DEBATE, OTRA;
}

public enum TipoInscripcion {

    ASISTENTE, CONFERENCISTA, TALLERISTA, OTRO;

}

public enum TipoPago {

    EFECTIVO, TRANSFERENCIA, TARJETA;
}
```

Clase para conectar la base de datos

```
public class ConectarDBA {

    private final String URL = "jdbc:mysql://localhost:3306/Triforce";
    private final String USER = "root";
    private final String PASSWORD = "010418";
    private Connection connection;

    public ConectarDBA() {
        connect();
    }

    public void connect() {
        try {
            if (connection == null || connection.isClosed()) {
                connection = DriverManager.getConnection(URL, USER, PASSWORD);
                System.out.println("Conectoooooo");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("No conectoooo");
        }
    }

    public Connection getConnect() {

        try {
            if (connection == null || connection.isClosed()) {
                connect();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }

    public void close() {
        if (connection != null) {
            try {
                connection.close();
                System.out.println("Se cerro conexion");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

}
```