

Universidad de San Carlos de Guatemala
Centro Universitario de Occidente -CUNOC-
Introducción a la Programación y Computación 2 “A”
Ing. José Moisés Granados

Manual técnico práctica 1

https://github.com/RonaldSamayoa/Practica1_IPC2_202031046.git

Estudiante

Ronald Renán Samayoa Martínez 202031046

Carné

Quetzaltenango 19 de agosto de 2025

INTRODUCCIÓN

El siguiente manual se ha desarrollado con la finalidad de dar a conocer la información necesaria para realizar mantenimiento, ejecución y exploración del código de la práctica 1, la cual consta de diferentes paquetes e interfaces gráficas para la ejecución de una aplicación de eventos.

El manual ofrece la información necesaria de ¿cómo está realizado el código? para que una persona (Desarrollador en el lenguaje Java con conocimientos de implementación de Parsers) que desee comprender y posteriormente editar el código lo haga de una manera apropiada, dando a conocer la estructura del desarrollo del aplicativo.

El manual técnico hace referencia a la información necesaria con el fin de orientar al personal en la concepción, planteamiento, análisis, programación e instalación del sistema. Es de notar que la redacción propia del manual técnico está orientada a personal con conocimientos en sistemas y tecnologías de informática y conocimientos de programación sobre el lenguaje Java. Para su desarrollo se ha decidido utilizar programación modular y la estructura modelo, vista y controlador (MVC) y el patrón de diseño Data Access Object (DAO) (como una capa intermedia entre la aplicación y la base de datos). También se ha utilizado la herramienta Swing para crear una interfaz gráfica más amigable con el usuario.



DESARROLLO

Para el desarrollo del código se utilizó un IDE, el cual se trata de NetBeans 25. Ademas fue compilado mediante un Java Development Kit, cuyo openjdk utiliza la version 24.0.1.

El código permite ser ejecutado desde la terminal de nuestro sistema operativo, ya sea la terminal de Linux o el uso del terminal Powershell en Windows.

Prerrequisitos

- Mínimo 4 GB de memoria RAM
- Sistema operativo, preferentemente Linux o Windows
- El archivo ejecutable se encuentra comprimido en .jar, por lo que al ejecutar debe utilizar el comando java -jar antes del nombre del archivo ejecutable

Utilizacion del programa

A continuacion se presenta como accesar en la terminal utizando el comando java -jar:

```
ronald@ronald-IdeaPad-3-14ITL6:~/NetBeansProjects/EventosHyrule/target$ java -jar EventosHyrule-1.0-SNAPSHOT-jar-with-dependencies.jar
```

NOTA: Dentro de la carpeta target se encuentran dos archivos generados, de los cuales el correcto es EventosHyrule-1.0-SNAPSHOT-jar-with-dependencies.jar, debido a que este contiene las dependencias necesarias para el funcionamiento completo de la aplicación.

ALGORITMOS DE MODELAMIENTO

Para el desarrollo del código del proyecto 2, se ha hecho el uso del patrón de arquitectura Modelo Vista Controlador (MVC), al igual que conceptos basicos de programacion modular y el patrón de diseño Data Acces Object (DAO). El archivo principal se encuentra enraizado en la carpeta de origen (AnalizadorSintactico).

Paquete Modelo

Clase Participante: Representa a una persona que puede registrarse o participar de un evento. Almacena informacion basica de un usuario del sistema, sirviendo como modelo de datos para gestionar a los participantes

```
public class Participante {  
    private String correo;  
    private String nombreCompleto;  
    private String tipoParticipante;  
    private String institucion;  
  
    public Participante() {}  
  
    public Participante(String nombreCompleto, String tipoParticipante, String institucion, String correo){  
        this.correo = correo;  
        this.nombreCompleto = nombreCompleto;  
        this.tipoParticipante = tipoParticipante;  
        this.institucion = institucion;  
    }  
}
```

Clase Evento: Representa un acontecimiento a realizar. Modela la informacion fundamental de un evento.

```
public class Evento {  
    private String codigoEvento;  
    private Date fecha; //para compatibilidad con la jdbc  
    private String tipoEvento;  
    private String titulo;  
    private String ubicacion;  
    private int cupoMaximo;  
    private BigDecimal costoInscripcion;  
  
    public Evento () {}  
  
    public Evento (String codigoEvento, Date fecha, String tipoEvento, String titulo, String ubicacion, int cupoMaximo, BigDecimal costoInscripcion){  
        this.codigoEvento = codigoEvento;  
        this.fecha = fecha;  
        this.tipoEvento = tipoEvento;  
        this.titulo = titulo;  
        this.ubicacion = ubicacion;  
        this.cupoMaximo = cupoMaximo;  
        this.costoInscripcion = costoInscripcion;  
    }  
}
```

Clase Actividad: Representa una reunión específica que forma parte de un evento mas grande. Permite gestionar las subdivisiones de un evento, vinculando a este mediante su código.

```
public class Actividad {  
    private String codigoActividad;  
    private String codigoEvento;  
    private String tipoActividad;  
    private String tituloActividad;  
    private String correoEncargado;  
    private Time horaInicio;  
    private Time horaFin;  
    private int cupoMaximo;  
  
    public Actividad () {}  
  
    public Actividad (String codigoActividad, String codigoEvento, String tipoActividad,  
        String tituloActividad, String correoEncargado, Time horaInicio, Time horaFin, int cupoMaximo) {  
        this.codigoActividad = codigoActividad;  
        this.codigoEvento = codigoEvento;  
        this.tipoActividad = tipoActividad;  
        this.tituloActividad = tituloActividad;  
        this.correoEncargado = correoEncargado;  
        this.horaInicio = horaInicio;  
        this.horaFin = horaFin;  
        this.cupoMaximo = cupoMaximo;  
    }  
}
```

Clase Inscripcion: Representa el registro de un participante a un evento específico.

```
public class Inscripcion {  
    private int idInscripcion;  
    private String correoParticipante;  
    private String codigoEvento;  
    private String tipoInscripcion; // asistente, conferencista, etc...  
    private boolean validada;  
  
    //Constructor vacio para crear un objeto y llenar uno a uno los atributos  
    public Inscripcion () {}  
  
    //Cnstructor para cuando se tienen todos los datos listos  
    public Inscripcion (int idInscripcion, String correoParticipante, String codigoEvento, String tipoInscripcion, boolean validada) {  
        this.idInscripcion = idInscripcion;  
        this.correoParticipante = correoParticipante;  
        this.codigoEvento = codigoEvento;  
        this.tipoInscripcion = tipoInscripcion;  
        this.validada = validada;  
    }  
}
```

Clase Pago: Representa una transaccion financiera asociada a la inscripción de un participante a un evento.

```
public class Pago {  
    private int idPago;  
    private String correoParticipante;  
    private String codigoEvento;  
    private String metodoPago;  
    private BigDecimal monto;  
  
    public Pago () {}  
  
    public Pago (int idPago, String correoParticipante, String codigoEvento, String metodoPago, BigDecimal monto){  
        this.idPago = idPago;  
        this.correoParticipante = correoParticipante;  
        this.codigoEvento = codigoEvento;  
        this.metodoPago = metodoPago;  
        this.monto = monto;  
    }  
}
```

Clase Asistencia: Registra la presencia de un participante en una actividad específica.

```
public class Asistencia {  
    private int idAsistencia;  
    private String correoParticipante;  
    private String codigoActividad;  
  
    public Asistencia(){}  
  
    public Asistencia(int idAsistencia, String correoParticipante, String codigoActividad){  
        this.idAsistencia = idAsistencia;  
        this.correoParticipante = correoParticipante;  
        this.codigoActividad = codigoActividad;  
    }  
}
```

Clase Certificado:Representa a un documento digital que acredita la participacion de una persona en un evento.

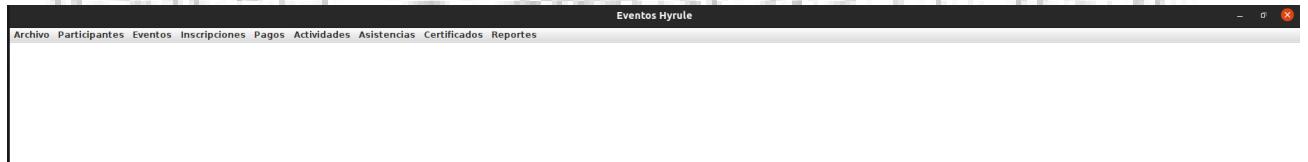
```
public class Certificado {  
    private int idCertificado;  
    private String correoParticipante;  
    private String codigoEvento;  
    private String rutaArchivo;  
  
    public Certificado() {}  
  
    public Certificado(int idCertificado, String correoParticipante, String codigoEvento, String rutaArchivo){  
        this.idCertificado = idCertificado;  
        this.correoParticipante = correoParticipante;  
        this.codigoEvento = codigoEvento;  
        this.rutaArchivo = rutaArchivo;  
    }  
}
```

Clase DBConnection: Clase utilitaria responsable de establecer y gestionar la conexión con la base de datos.

```
public class DBConnection {  
    private static final String IP = "localhost";  
    private static final int PUERTO = 3306;  
    private static final String SCHEMA = "eventos_hyrule";  
    private static final String USER_NAME = "admin";  
    private static final String PASSWORD = "Ronald01!";  
    private static final String URL = "jdbc:mysql://" +  
        IP + ":" + PUERTO + "/" + SCHEMA  
        + "?useSSL=false&serverTimezone=UTC";  
  
    public Connection getConnection() throws SQLException{  
        return DriverManager.getConnection(URL, USER_NAME, PASSWORD);  
    }  
  
    //probar conexión manualmente  
    public void connect() {  
        System.out.println("URL de conexión: " + URL);  
  
        try (Connection connection = DriverManager.getConnection(URL, USER_NAME, PASSWORD);){  
            System.out.println("Esquema: " + connection.getSchema());  
            System.out.println("Catalogo: " + connection.getCatalog());  
  
        } catch (SQLException e) {  
            System.out.println("Error al conectar");  
            e.printStackTrace();  
        }  
    }  
}
```

Paquete Vista

Propósito: Establecer una interfaz gráfica para el manejo de la aplicación MainFrame



Paquete DAO

Clase ParticipanteDAO: Objeto de acceso a datos para Participante. Representa a los usuarios del sistema y administra el registro y la gestión de todas las personas que interactúan con el sistema.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java x InscripcionControlador.java x ParticipanteControlador.java x
Projects x Files Services Source History <default config> 537/4934.0MB Search (Ctrl+)
com.mycompany.eventoshyrule.c
  AsistenciaController.java
  CertificadoController.java
  EventoController.java
  IncripcionController.java
  PagoController.java
  ParticipanteController.java
  ReporteController.java
  com.mycompany.eventoshyrule.c
    EventoDAO.java
    AsistenciaDAO.java
    CertificadoDAO.java
    EventoDAO.java
    IncripcionDAO.java
    PagoDAO.java
    ParticipanteDAO.java
  com.mycompany.eventoshyrule.r
    Actividad.java
    Asistencia.java
    Certificado.java
    DBConnection.java
    Evento.java
    Incripcion.java
    Pago.java
    Participante.java
  com.mycompany.eventoshyrule.p
    Parser.java
  com.mycompany.eventoshyrule.v
    FrmActividad.java
    FrmAsistencia.java
    FrmCertificado.java
    FrmEvento.java
    FrmIncripcion.java
    FrmListaActividades.java
    FrmListaEventos.java
    FrmListaParticipantes.java
    FrmPago.java
    FrmReporte.java
    MainFrame.java
  Test Packages
  Dependencies
  Java Dependencies
  Project Files
  juegoBasket1
  Output - Run (EventosHyrule) x
  >>
  104:5 INS Unix (LF)

```

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java x InscripcionControlador.java x ParticipanteControlador.java x
Projects x Files Services Source History <default config> 205/2934.0MB Search (Ctrl+)
com.mycompany.eventoshyrule.c
  AsistenciaController.java
  CertificadoController.java
  EventoController.java
  IncripcionController.java
  PagoController.java
  ParticipanteController.java
  ReporteController.java
  com.mycompany.eventoshyrule.c
    EventoDAO.java
    AsistenciaDAO.java
    CertificadoDAO.java
    EventoDAO.java
    IncripcionDAO.java
    PagoDAO.java
    ParticipanteDAO.java
  com.mycompany.eventoshyrule.r
    Actividad.java
    Asistencia.java
    Certificado.java
    DBConnection.java
    Evento.java
    Incripcion.java
    Pago.java
    Participante.java
  com.mycompany.eventoshyrule.p
    Parser.java
  com.mycompany.eventoshyrule.v
    FrmActividad.java
    FrmAsistencia.java
    FrmCertificado.java
    FrmEvento.java
    FrmIncripcion.java
    FrmListaActividades.java
    FrmListaEventos.java
    FrmListaParticipantes.java
    FrmPago.java
    FrmReporte.java
    MainFrame.java
  Test Packages
  Dependencies
  Java Dependencies
  Project Files
  juegoBasket1
  Output - Run (EventosHyrule) x
  >>
  33:54 INS Unix (LF)

```

Clase EventoDAO: Objeto de acceso a datos para Evento. Gestiona la persistencia de todos los eventos. Implementa las operaciones CRUD.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java x InscripcionControlador.java x ParticipanteControlador.java x
Projects x Files Services Source History ...va EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java x InscripcionControlador.java x ParticipanteControlador.java x
Source
public boolean insertar(Evento evento) {
    String sql = "INSERT INTO Evento (codigo_evento, fecha, tipo_evento, titulo, ubicacion, cupo_maximo, costo_inscripcion) "
    try (Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, evento.getCodigoEvento());
        ps.setDate(2, evento.getFecha());
        ps.setString(3, evento.getTipoEvento());
        ps.setString(4, evento.getTitulo());
        ps.setInt(5, evento.getUbicacion());
        ps.setInteger(6, evento.getCupoMaximo());
        ps.setBigDecimal(7, evento.getCostoInscripcion());
        return ps.executeUpdate() > 0;
    } catch (SQLException e) {
        System.out.println("Error al insertar evento: " + e.getMessage());
        return false;
    }
}

// listar todos los eventos
public List<Evento> ListarTodos() {
    List<Evento> lista = new ArrayList<>();
    String sql = "SELECT * FROM Evento";
    try (Connection conn = dbConnection.getConnection();
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql)) {
        while (rs.next()) {
            Evento e = new Evento(
                rs.getString("codigo_evento"),
                rs.getDate("fecha"),
                rs.getString("tipo_evento"),
                rs.getString("titulo"),
                rs.getString("ubicacion"),
                rs.getInt("cupo_maximo"),
                rs.getBigDecimal("costo_inscripcion")
            );
            lista.add(e);
        }
    } catch (SQLException e) {
        System.out.println("Error al listar eventos: " + e.getMessage());
    }
}

```

Output - Run (EventosHyrule) x

109:26 INS Unix (LF)

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java x InscripcionControlador.java x ParticipanteControlador.java x
Projects x Files Services Source History ...va EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java x InscripcionControlador.java x ParticipanteControlador.java x
Source
// busqueda por codigo
public Evento buscarPorCodigo(String codigo) {
    String sql = "SELECT * FROM Evento WHERE Codigo_evento = ?";
    try (Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setString(1, codigo);
        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                return new Evento(
                    rs.getString("codigo_evento"),
                    rs.getDate("fecha"),
                    rs.getString("tipo_evento"),
                    rs.getString("titulo"),
                    rs.getString("ubicacion"),
                    rs.getInt("cupo_maximo"),
                    rs.getBigDecimal("costo_inscripcion")
                );
            }
        } catch (SQLException e) {
            System.out.println("Error al buscar evento: " + e.getMessage());
        }
        return null;
    }
}

// actualizaciones
public boolean actualizar(Evento evento) {
    String sql = "UPDATE Evento SET fecha = ?, tipo_evento = ?, titulo = ?, ubicacion = ?, cupo_maximo = ?, costo_inscripcion = ? "
    "WHERE Codigo_evento = ?";
    try (Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setDate(1, evento.getFecha());
        ps.setString(2, evento.getTipoEvento());
        ps.setString(3, evento.getTitulo());
        ps.setString(4, evento.getUbicacion());
        ps.setInt(5, evento.getCupoMaximo());
        ps.setBigDecimal(6, evento.getCostoInscripcion());
        ps.setString(7, evento.getCodigoEvento());
        return ps.executeUpdate() > 0;
    } catch (SQLException e) {
        System.out.println("Error al actualizar evento: " + e.getMessage());
        return false;
    }
}

```

Output - Run (EventosHyrule) x

109:26 INS Unix (LF)

Clase ActividadDAO: Objeto de acceso a datos para Actividad. Se encarga de las operaciones CRUD relacionadas con las actividades en la base de datos

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java x AsistenciaDAO.java x EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java...
Projects x Files Services
Source History ...
ActividadDAO.java
public boolean insertar (Actividad actividad) {
    String sql = "INSERT INTO Actividad (codigo_actividad, codigo_evento, tipo_actividad, titulo_actividad, correo_encargado, hora_inicio, hora_fin, cupo_maximo)";
    try{Connection conn = dbConnection.getConnection ();
        PreparedStatement ps = conn.prepareStatement(sql)} // para ejecutar los parametros query de entrada
        //ps.setString(1, actividad.getCodigoActividad());
        ps.setString(2, actividad.getCodigoEvento());
        ps.setString(3, actividad.getTipoActividad());
        ps.setString(4, actividad.getTituloActividad());
        ps.setString(5, actividad.getCorreoEncargado());
        ps.setDouble(6, actividad.getHoraInicio());
        ps.setTime(7, actividad.getHoraFin());
        ps.setInt(8, actividad.getCupoMaximo());
    }
    return ps.executeUpdate(); //ejecuta la insercion y retorna true si fue exitosa
} catch (SQLException e){
    System.out.println("Error al registrar actividad: " + e.getMessage());
    return false; //false en caso de error
}

//Listar datos (select * from)
public List<Actividad> listarTodos(){
    List<Actividad> lista = new ArrayList<>();
    String sql = "SELECT * FROM Actividad";
    try{(Connection conn = dbConnection.getConnection ());
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql)} //ejecuta query y obtiene resultados
    while (rs.next()){
        Actividad a = new Actividad // crea un objeto a partir de los dato de la fila actual
        rs.getString("codigo_actividad");
        rs.getString("codigo_evento");
        rs.getString("tipo_actividad");
        rs.getString("titulo_actividad");
        rs.getString("correo_encargado");
        rs.getTime("hora_inicio");
        rs.getTime("hora_fin");
        rs.getInt("cupo_maximo");
    }
    lista.add(a);
}
} catch (SQLException e) {
    System.out.println("Error al listar todos los datos: " + e.getMessage());
    return lista;
}
}

```

Output - Run (EventosHrule) x

15:18 INS Unix (LF)

EventosHrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java x AsistenciaDAO.java x EventoDAO.java x ParticipanteDAO.java x PagoDAO.java x InscripcionDAO.java x CertificadoDAO.java x EventoControlador.java...
Projects x Files Services
Source History ...
ActividadDAO.java
public boolean insertar (Actividad actividad) {
    String sql = "INSERT INTO Actividad (codigo_actividad, codigo_evento, tipo_actividad, titulo_actividad, correo_encargado, hora_inicio, hora_fin, cupo_maximo)";
    try{Connection conn = dbConnection.getConnection ();
        PreparedStatement ps = conn.prepareStatement(sql)} // para ejecutar los parametros query de entrada
        //ps.setString(1, actividad.getCodigoActividad());
        ps.setString(2, actividad.getCodigoEvento());
        ps.setString(3, actividad.getTipoActividad());
        ps.setString(4, actividad.getTituloActividad());
        ps.setString(5, actividad.getCorreoEncargado());
        ps.setDouble(6, actividad.getHoraInicio());
        ps.setTime(7, actividad.getHoraFin());
        ps.setInt(8, actividad.getCupoMaximo());
    }
    return ps.executeUpdate(); //ejecuta la insercion y retorna true si fue exitosa
} catch (SQLException e){
    System.out.println("Error al registrar actividad: " + e.getMessage());
    return false; //false en caso de error
}

//Listar datos (select * from)
public List<Actividad> listarTodos(){
    List<Actividad> lista = new ArrayList<>();
    String sql = "SELECT * FROM Actividad";
    try{(Connection conn = dbConnection.getConnection ());
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql)} //ejecuta query y obtiene resultados
    while (rs.next()){
        Actividad a = new Actividad // crea un objeto a partir de los dato de la fila actual
        rs.getString("codigo_actividad");
        rs.getString("codigo_evento");
        rs.getString("tipo_actividad");
        rs.getString("titulo_actividad");
        rs.getString("correo_encargado");
        rs.getTime("hora_inicio");
        rs.getTime("hora_fin");
        rs.getInt("cupo_maximo");
    }
    lista.add(a);
}
} catch (SQLException e) {
    System.out.println("Error al listar todos los datos: " + e.getMessage());
    return lista;
}
}

```

Output - Run (EventosHrule) x

15:18 INS Unix (LF)

Clase InscripcionDAO: Objeto de acceso a datos para Inscripcion. Es el nucleo de interaccion entre participante y evento. Controla el proceso de registro y es fundamental para gestionar el cupo de los eventos y el estado de participaciones.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> 512.951.184MB Search (Ctrl+)
Projects Files Services ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
Source History <...> ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
21 //Insertar datos
22 public boolean insertar (Incripcion inscripcion) {
23     String sql = "INSERT INTO Incripcion (correo_participante, codigo_evento, tipo_incripcion, validada)"
24     try( Connection conn = dbConnection.getConnection();
25         PreparedStatement ps = conn.prepareStatement(sql)) {
26
27         ps.setString(1,inscripcion.getCorreoParticipante());
28         ps.setString(2,inscripcion.getCodigoEvento());
29         ps.setString(3,inscripcion.getTipoIncripcion());
30         ps.setBoolean(4,inscripcion.isValidada());
31
32         return ps.executeUpdate() > 0;
33     } catch (SQLException e) {
34         System.out.println("Error al registrar la inscripcion: " + e.getMessage());
35         return false;
36     }
37 }
38
39 //Listar datos (select * from)
40 public List<Incripcion> listarTodos(){
41     List<Incripcion> lista = new ArrayList<>();
42     String sql = "SELECT * FROM Incripcion";
43     try (Connection conn = dbConnection.getConnection();
44         Statement st = conn.createStatement();
45         ResultSet rs = st.executeQuery(sql)) {
46
47         while (rs.next()) {
48             Incripcion i = new Incripcion(
49                 rs.getInt("id_incripcion"),
50                 rs.getString("correo_participante"),
51                 rs.getString("codigo_evento"),
52                 rs.getString("tipo_incripcion"),
53                 rs.getBoolean("validada")
54             );
55             lista.add(i);
56         }
57     } catch (SQLException e) {
58         System.out.println("Error al listar inscripciones: " + e.getMessage());
59     }
60     return lista;
61 }
62
63 //busqueda por id
64
65
Output - Run (EventosHrule) >
>>

```

137:30 INS Unix (LF)

EventosHrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config> 512.951.184MB Search (Ctrl+)
Projects Files Services ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
Source History <...> ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
21 //Insertar datos
22 public boolean insertar (Incripcion inscripcion) {
23     String sql = "INSERT INTO Incripcion (correo_participante, codigo_evento, tipo_incripcion, validada)"
24     try( Connection conn = dbConnection.getConnection();
25         PreparedStatement ps = conn.prepareStatement(sql)) {
26
27         ps.setString(1,inscripcion.getCorreoParticipante());
28         ps.setString(2,inscripcion.getCodigoEvento());
29         ps.setString(3,inscripcion.getTipoIncripcion());
30         ps.setBoolean(4,inscripcion.isValidada());
31
32         return ps.executeUpdate() > 0;
33     } catch (SQLException e) {
34         System.out.println("Error al registrar la inscripcion: " + e.getMessage());
35         return false;
36     }
37 }
38
39 //Listar datos (select * from)
40 public List<Incripcion> listarTodos(){
41     List<Incripcion> lista = new ArrayList<>();
42     String sql = "SELECT * FROM Incripcion";
43     try (Connection conn = dbConnection.getConnection();
44         Statement st = conn.createStatement();
45         ResultSet rs = st.executeQuery(sql)) {
46
47         while (rs.next()) {
48             Incripcion i = new Incripcion(
49                 rs.getInt("id_incripcion"),
50                 rs.getString("correo_participante"),
51                 rs.getString("codigo_evento"),
52                 rs.getString("tipo_incripcion"),
53                 rs.getBoolean("validada")
54             );
55             lista.add(i);
56         }
57     } catch (SQLException e) {
58         System.out.println("Error al listar inscripciones: " + e.getMessage());
59     }
60     return lista;
61 }
62
63 //busqueda por id
64
65
Output - Run (EventosHrule) >
>>

```

137:30 INS Unix (LF)

Clase PagoDAO: Objeto de acceso a datos para Pago. Se encarga de registrar y consultar los pagos realizados por los participantes.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
Projects x Files Services ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
Source History <default config> Search (Ctrl+)
512.9M 1384M
1. Insertar datos
public boolean insertar (Incripcion inscripcion) {
    String sql = "INSERT INTO Inscripcion (correo_participante, codigo_evento, tipo_incripcion, validada)";
    try( Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setString(1,inscripcion.getCorreoParticipante());
        ps.setInt(2,inscripcion.getCodigoEvento());
        ps.setString(3,inscripcion.getTipoIncripcion());
        ps.setBoolean(4,inscripcion.isValidada());
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            return true;
        } else {
            System.out.println("Error al registrar la inscripcion: " + e.getMessage());
            return false;
        }
    }
}

//Listar datos (select * from)
public List<Incripcion> listarTodos(){
    List<Incripcion> lista = new ArrayList<>();
    String sql = "SELECT * FROM Inscripcion";
    try( Connection conn = dbConnection.getConnection();
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql)){
        while (rs.next()){
            Incripcion i = new Incripcion(
                rs.getInt("id_incripcion"),
                rs.getString("correo_participante"),
                rs.getInt("codigo_evento"),
                rs.getString("tipo_incripcion"),
                rs.getBoolean("validada")
            );
            lista.add(i);
        }
    } catch (SQLException e) {
        System.out.println("Error al listar inscripciones: " + e.getMessage());
    }
    return lista;
}

//busqueda por id
public Pago buscarPorId(int id){
    String sql = "SELECT * FROM Pago WHERE id_pago = ?";
    try( Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setInt(1, id);
        try(ResultSet rs = ps.executeQuery()){
            if (rs.next()){
                return new Pago(
                    rs.getInt("id_pago"),
                    rs.getString("correo_participante"),
                    rs.getInt("codigo_evento"),
                    rs.getString("metodo_pago"),
                    rs.getBigDecimal("monto")
                );
            }
        } catch (SQLException e) {
            System.out.println("Error al buscar pago: " + e.getMessage());
        }
    } catch (SQLException e) {
        System.out.println("Error al buscar pago: " + e.getMessage());
    }
    return null;
}

//actualizar pago
public boolean actualizar(Pago pago){
    String sql = "UPDATE Pago SET correo_participante = ?, codigo_evento = ?, monto = ? "
    + "WHERE id_pago = ?";
    try( Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setString(1, pago.getCorreoParticipante());
        ps.setInt(2, pago.getCodigoEvento());
        ps.setBigDecimal(3, pago.getMonto());
        ps.setInt(4, pago.getIdPago());
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            return true;
        } else {
            System.out.println("Error al actualizar pago: " + e.getMessage());
            return false;
        }
    } catch (SQLException e) {
        System.out.println("Error al actualizar pago: " + e.getMessage());
    }
}

```

Output - Run (EventosHyrule) x

137:30 INS Unix (LF)

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
Projects x Files Services ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java IncripcionDAO.java CertificadoDAO.java EventoControlador.java
Source History <default config> Search (Ctrl+)
644M 129MB
1. Insertar datos
public boolean insertar (Incripcion inscripcion) {
    String sql = "INSERT INTO Inscripcion (correo_participante, codigo_evento, tipo_incripcion, validada)";
    try( Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setString(1,inscripcion.getCorreoParticipante());
        ps.setInt(2,inscripcion.getCodigoEvento());
        ps.setString(3,inscripcion.getTipoIncripcion());
        ps.setBoolean(4,inscripcion.isValidada());
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            return true;
        } else {
            System.out.println("Error al registrar la inscripcion: " + e.getMessage());
            return false;
        }
    }
}

//Listar datos (select * from)
public List<Incripcion> listarTodos(){
    List<Incripcion> lista = new ArrayList<>();
    String sql = "SELECT * FROM Inscripcion";
    try( Connection conn = dbConnection.getConnection();
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(sql)){
        while (rs.next()){
            Incripcion i = new Incripcion(
                rs.getInt("id_incripcion"),
                rs.getString("correo_participante"),
                rs.getInt("codigo_evento"),
                rs.getString("tipo_incripcion"),
                rs.getBoolean("validada")
            );
            lista.add(i);
        }
    } catch (SQLException e) {
        System.out.println("Error al listar inscripciones: " + e.getMessage());
    }
    return lista;
}

//busqueda por id
public Pago buscarPorId(int id){
    String sql = "SELECT * FROM Pago WHERE id_pago = ?";
    try( Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setInt(1, id);
        try(ResultSet rs = ps.executeQuery()){
            if (rs.next()){
                return new Pago(
                    rs.getInt("id_pago"),
                    rs.getString("correo_participante"),
                    rs.getInt("codigo_evento"),
                    rs.getString("metodo_pago"),
                    rs.getBigDecimal("monto")
                );
            }
        } catch (SQLException e) {
            System.out.println("Error al buscar pago: " + e.getMessage());
        }
    } catch (SQLException e) {
        System.out.println("Error al buscar pago: " + e.getMessage());
    }
    return null;
}

//actualizar pago
public boolean actualizar(Pago pago){
    String sql = "UPDATE Pago SET correo_participante = ?, codigo_evento = ?, monto = ? "
    + "WHERE id_pago = ?";
    try( Connection conn = dbConnection.getConnection();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setString(1, pago.getCorreoParticipante());
        ps.setInt(2, pago.getCodigoEvento());
        ps.setBigDecimal(3, pago.getMonto());
        ps.setInt(4, pago.getIdPago());
        int rowsAffected = ps.executeUpdate();
        if (rowsAffected > 0) {
            return true;
        } else {
            System.out.println("Error al actualizar pago: " + e.getMessage());
            return false;
        }
    } catch (SQLException e) {
        System.out.println("Error al actualizar pago: " + e.getMessage());
    }
}

```

Output - Run (EventosHyrule) x

113:28 INS Unix (LF)

Clase AsistenciaDAO: Objeto de acceso a datos para Asistencia. Gestiona la persistencia de los registros de asistencia a las actividades.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java EventoControlador.java
Projects x Files Services Source History <default config> 693.1/1309MB Search (Ctrl+)
AsistenciaControlador.java
AsistenciaDAO.java
EventoControlador.java
InscripcionControlador.java
PagoControlador.java
ParticipanteControlador.java
ReporteControlador.java
com.mycompany.eventoshyrule.c
AsistenciaDAO.java
Asistencia.java
Certificado.java
DBConnection.java
Evento.java
Inscripcion.java
Pago.java
Participante.java
com.mycompany.eventoshyrule.e
Actividad.java
Asistencia.java
Certificado.java
DBConnection.java
Evento.java
Inscripcion.java
Pago.java
Participante.java
com.mycompany.eventoshyrule.f
Parser.java
com.mycompany.eventoshyrule.v
FrmActividad.java
FrmAsistencia.java
FrmCertificado.java
FrmEvento.java
FrmInscripcion.java
FrmListaActividades.java
FrmListarEventos.java
FrmListarParticipantes.java
FrmPago.java
FrmParticipante.java
FrmReportes.java
MainFrame.java
Test Packages
Dependencies
Java Dependencies
Project Files
juegoBasket1
Source - AsistenciaDAO.java
...
19 public AsistenciaDAO() {
20     this.dbConnection = new DBConnection();
21 }
22
23
24
25
26
27 //insertar datos
28 public boolean insertar (Asistencia asistencia) {
29     String sql = "INSERT INTO Asistencia (correo_participante, codigo_actividad)"
30             + "VALUES (?, ?)";
31     try( Connection conn = dbConnection.getConnection ();
32         PreparedStatement ps = conn.prepareStatement(sql)){ //para ejecutar los parametros query de entrada
33         ps.setString(1,asistencia.getCorreoParticipante());
34         ps.setString(2,asistencia.getCodigoActividad());
35
36         return ps.executeUpdate()>0; //ejecuta la insercion y retorna true si fue exitosa
37     } catch (SQLException e){
38         System.out.println("Error al registrar asistencia: " + e.getMessage());
39         return false;
40     }
41
42
43
44 //Listado de asistencias
45 public List<Asistencia> listarTodos(){
46     List<Asistencia> lista = new ArrayList<>();
47     String sql = "SELECT * FROM Asistencia";
48     try ( Connection conn = dbConnection.getConnection();
49          Statement st = conn.createStatement();
50          ResultSet rs = st.executeQuery(sql)){ //ejecuta query y obtiene resultados
51
52         while (rs.next()){
53             Asistencia a = new Asistencia(
54                 rs.getInt("id_asistencia"),
55                 rs.getString("correo_participante"),
56                 rs.getString("codigo_actividad")
57             );
58             lista.add(a);
59         }
60     } catch (SQLException e){
61         System.out.println("Error al listar asistencias: " + e.getMessage());
62     }
63     return lista;
64 }
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

Output - Run (EventosHyrule) >

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java EventoControlador.java
Projects x Files Services Source History <default config> 693.4/1309MB Search (Ctrl+)
AsistenciaControlador.java
AsistenciaDAO.java
EventoControlador.java
InscripcionControlador.java
PagoControlador.java
ParticipanteControlador.java
ReporteControlador.java
com.mycompany.eventoshyrule.c
AsistenciaDAO.java
Asistencia.java
Certificado.java
DBConnection.java
Evento.java
Inscripcion.java
Pago.java
Participante.java
com.mycompany.eventoshyrule.e
Actividad.java
Asistencia.java
Certificado.java
DBConnection.java
Evento.java
Inscripcion.java
Pago.java
Participante.java
com.mycompany.eventoshyrule.f
Parser.java
com.mycompany.eventoshyrule.v
FrmActividad.java
FrmAsistencia.java
FrmCertificado.java
FrmEvento.java
FrmInscripcion.java
FrmListaActividades.java
FrmListarEventos.java
FrmListarParticipantes.java
FrmPago.java
FrmParticipante.java
FrmReportes.java
MainFrame.java
Test Packages
Dependencies
Java Dependencies
Project Files
juegoBasket1
Source - AsistenciaDAO.java
...
61
62
63 //busqueda por id
64 public Asistencia buscarPorId(int id){
65     String sql = "SELECT * FROM Asistencia WHERE id_asistencia = ?";
66     try ( Connection conn = dbConnection.getConnection();
67         PreparedStatement ps = conn.prepareStatement(sql)){
68
69         ps.setInt(1, id);
70         try (ResultSet rs = ps.executeQuery()){
71             if (rs.next()){
72                 Asistencia(
73                     rs.getInt("id_asistencia"),
74                     rs.getString("correo_participante"),
75                     rs.getString("codigo_actividad")
76                 );
77             }
78         } catch (SQLException e){
79             System.out.println("Error al buscar asistencia: " + e.getMessage());
80         }
81         return null;
82     }
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

Output - Run (EventosHyrule) >

Clase CertificadoDAO: Objeto de acceso a datos para Certificado. Mantiene un registro de que certificados existen, para que evento y participante fue emitido y donde se almacena el archivo.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java EventoControlador.java...
Projects x Files Services ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java...
Source History ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java...
public boolean insertar (Certificado certificado) {
    String sql = "INSERT INTO Certificado (correo_participante, codigo_evento, ruta_archivo)"
        + "VALUES (?, ?, ?)";
    try( Connection conn = dbConnection.getConnection ();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setString(1,certificado.getCorreoParticipante());
        ps.setString(2,certificado.getCodigoEvento());
        ps.setString(3,certificado.getRutaArchivo());
        return ps.executeUpdate() > 0;
    } catch (SQLException e){
        System.out.println("Error al registrar certificado: " + e.getMessage());
        return false;
    }
}

//Listar datos (select * from)
public List <Certificado> listarTodos(){
    List<Certificado> lista = new ArrayList<>();
    String sql = "SELECT * FROM Certificado";
    try ( Connection conn = dbConnection.getConnection ();
        Statement st = conn.createStatement ();
        ResultSet rs = st.executeQuery(sql)){
        while (rs.next()){
            Certificado c = new Certificado(
                rs.getInt("id_certificado"),
                rs.getString("correo_participante"),
                rs.getString("codigo_evento"),
                rs.getString("titulo_actividad")
            );
            lista.add(c);
        }
    } catch (SQLException e){
        System.out.println("Error al listar certificados: " + e.getMessage());
    }
    return lista;
}

//busqueda por id
public Certificado buscarPorId(int id){
    String sql = "SELECT * FROM Certificado WHERE id_certificado = ?";
    try ( Connection conn = dbConnection.getConnection ();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ...
    }
}

```

Output - Run (EventosHyrule) x

100:7 INS Unix (LF)

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java EventoControlador.java...
Projects x Files Services ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java...
Source History ...va ActividadDAO.java AsistenciaDAO.java EventoDAO.java ParticipanteDAO.java PagoDAO.java InscripcionDAO.java CertificadoDAO.java...
public boolean insertar (Certificado certificado) {
    String sql = "INSERT INTO Certificado (correo_participante, codigo_evento, ruta_archivo)"
        + "VALUES (?, ?, ?)";
    try( Connection conn = dbConnection.getConnection ();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ps.setString(1,certificado.getCorreoParticipante());
        ps.setString(2,certificado.getCodigoEvento());
        ps.setString(3,certificado.getRutaArchivo());
        return ps.executeUpdate() > 0;
    } catch (SQLException e){
        System.out.println("Error al registrar certificado: " + e.getMessage());
        return false;
    }
}

//Listar datos (select * from)
public List <Certificado> listarTodos(){
    List<Certificado> lista = new ArrayList<>();
    String sql = "SELECT * FROM Certificado";
    try ( Connection conn = dbConnection.getConnection ();
        Statement st = conn.createStatement ();
        ResultSet rs = st.executeQuery(sql)){
        while (rs.next()){
            Certificado c = new Certificado(
                rs.getInt("id_certificado"),
                rs.getString("correo_participante"),
                rs.getString("codigo_evento"),
                rs.getString("titulo_actividad")
            );
            lista.add(c);
        }
    } catch (SQLException e){
        System.out.println("Error al listar certificados: " + e.getMessage());
    }
    return lista;
}

//busqueda por id
public Certificado buscarPorId(int id){
    String sql = "SELECT * FROM Certificado WHERE id_certificado = ?";
    try ( Connection conn = dbConnection.getConnection ();
        PreparedStatement ps = conn.prepareStatement(sql)){
        ...
    }
}

```

Output - Run (EventosHyrule) x

100:7 INS Unix (LF)

Paquete Controlador

Clase ActividadControlador: Gestiona las operaciones relacionadas con las actividades de eventos, implementa la logica de negocio para garantizar que una actividad sea valida antes de crearla.



Apache NetBeans IDE 20

File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects x Files Services

...va ActividadControlador.java x AsistenciaControlador.java x CertificadoControlador.java x Parser.java x MainFrame.java x pom.xml [Practica01] x FrmParticipante.java x FrmEvent.. x

Source History

ActividadControlador.java

```
public ActividadControlador(){
    this.actividadDAO = new ActividadDAO();
    this.eventoDAO = new EventoDAO();
    this.participanteDAO = new ParticipanteDAO();
    this.inscripcionDAO = new InscripcionDAO();
}

public boolean registrarActividad (String codigoActividad, String codigoEvento, String tipoActividad, String tituloActividad,
        String correoEncargado, Time horainicio, Time horaFin, int cupoMaximo){

    //comprobar que el evento existe
    Evento evento = eventoDAO.buscarPorCodigo(codigoEvento);
    if (evento == null) {
        System.out.println("No existe un evento con el codigo: " + codigoEvento);
        return false;
    }

    //validar que el participante encargado exista
    Participante encargado = participanteDAO.buscarPorCorreo(correoEncargado);
    if (encargado == null) {
        System.out.println("No existe un participante con el correo: " + correoEncargado);
        return false;
    }

    // validar la inscripcion del encargado en el evento
    Inscripcion inscripcionEncargado = inscripcionDAO.buscarPorCorreoEvento(correoEncargado, codigoEvento);
    if (inscripcionEncargado == null) {
        System.out.println("El encargado no esta inscrito");
        return false;
    }

    // el encargado no puede ser asistente
    if ("ASISTENTE".equalsIgnoreCase(inscripcionEncargado.getTipoInscripcion())){
        System.out.println("El encargado no puede ser asistente");
        return false;
    }

    //registrar la actividad
    Actividad actividad = new Actividad(codigoActividad, codigoEvento, tipoActividad, tituloActividad,
            correoEncargado, horainicio, horaFin, cupoMaximo);
    return actividadDAO.insertar(actividad);
}

//listar todas las actividades
public List<Actividad> listarActividades(){
    return actividadDAO.listarTodos();
}

// buscar actividad por codigo
public Actividad buscarActividad(String codigoActividad){
    return actividadDAO.buscarPorCodigo(codigoActividad);
}

//actualizacion de actividad
public boolean actualizarActividad(Actividad actividad){
    return actividadDAO.actualizar(actividad);
}

//eliminar a un participante
public boolean eliminarActividad(String codigoActividad){
    return actividadDAO.eliminar(codigoActividad);
}
```

Clase AsistenciaControlador: Gestiona el registro de asistencias a las actividades. Registra la asistencia de manera valida.

EventosHyrule - Apache NetBeans IDE 20

```
File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config...> 523.4108MB Search (Ctrl+)
```

Projects x Files Services

...va ActividadControlador.java AsistenciaControlador.java CertificadoControlador.java Parser.java MainFrame.java pom.xml [Practica01] FrmParticipante.java FrmEvent...

```
public class AsistenciaControlador {
    private AsistenciaDAO asistenciaDAO;
    private ParticipanteDAO participanteDAO;
    private ActividadDAO actividadDAO;

    public AsistenciaControlador() {
        this.asistenciaDAO = new AsistenciaDAO();
        this.participanteDAO = new ParticipanteDAO();
        this.actividadDAO = new ActividadDAO();
    }

    public boolean registrarAsistencia(String correoParticipante, String codigoActividad) {
        //validar que hay un participante
        Participante participante = participanteDAO.buscarPorCorreo(correoParticipante);
        if (participante == null) {
            System.out.println("No existe un participante con el correo: " + correoParticipante);
            return false;
        }

        //validar actividad
        Actividad actividad = actividadDAO.buscarPorCodigo(codigoActividad);
        if (actividad == null) {
            System.out.println("No existe actividad con codigo: " + codigoActividad);
            return false;
        }

        //registrar asistencia
        Asistencia asistencia = new Asistencia(0, correoParticipante, codigoActividad);
        return asistenciaDAO.insertar(asistencia);
    }

    //listar asistencias
    public List<Asistencia> listarAsistencias() {
        return asistenciaDAO.listarTodos();
    }

    //busqueda por id
    public Asistencia buscarAsistencia(int idAsistencia) {
        return asistenciaDAO.buscarPorId(idAsistencia);
    }

    //eliminar
    public boolean eliminarAsistencia(int idAsistencia) {
        return asistenciaDAO.eliminar(idAsistencia);
    }
}
```

42:16 INS Unix (LF)

Clase CertificadoControlador: Maneja la generación y gestión de certificados de participación.

EventosHyrule - Apache NetBeans IDE 20

```
File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config...> 523.4108MB Search (Ctrl+)
```

Projects x Files Services

...va ActividadControlador.java AsistenciaControlador.java CertificadoControlador.java Parser.java MainFrame.java pom.xml [Practica01] FrmParticipante.java FrmEvent...

```
public class AsistenciaControlador {
    private AsistenciaDAO asistenciaDAO;
    private ParticipanteDAO participanteDAO;
    private ActividadDAO actividadDAO;

    public AsistenciaControlador() {
        this.asistenciaDAO = new AsistenciaDAO();
        this.participanteDAO = new ParticipanteDAO();
        this.actividadDAO = new ActividadDAO();
    }

    public boolean registrarAsistencia(String correoParticipante, String codigoActividad) {
        //validar que hay un participante
        Participante participante = participanteDAO.buscarPorCorreo(correoParticipante);
        if (participante == null) {
            System.out.println("No existe un participante con el correo: " + correoParticipante);
            return false;
        }

        //validar actividad
        Actividad actividad = actividadDAO.buscarPorCodigo(codigoActividad);
        if (actividad == null) {
            System.out.println("No existe actividad con codigo: " + codigoActividad);
            return false;
        }

        //registrar asistencia
        Asistencia asistencia = new Asistencia(0, correoParticipante, codigoActividad);
        return asistenciaDAO.insertar(asistencia);
    }

    //listar asistencias
    public List<Asistencia> listarAsistencias() {
        return asistenciaDAO.listarTodos();
    }

    //busqueda por id
    public Asistencia buscarAsistencia(int idAsistencia) {
        return asistenciaDAO.buscarPorId(idAsistencia);
    }

    //eliminar
    public boolean eliminarAsistencia(int idAsistencia) {
        return asistenciaDAO.eliminar(idAsistencia);
    }
}
```

42:16 INS Unix (LF)

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va ActividadControlador.java AsistenciaControlador.java CertificadoControlador.java Parser.java MainFrame.java pom.xml [Practica01] FrmParticipante.java FrmEvento.java
Source History ...va ActividadControlador.java AsistenciaControlador.java CertificadoControlador.java Parser.java MainFrame.java pom.xml [Practica01] FrmParticipante.java FrmEvento.java
523.45108MB Search (Ctrl+F)
Projects x Files Services
> CalculadoraConPlas
> ejercicioNumerosAleatorios
> EventosHyrule [app-funcional-1]
  > Source Packages
    > com.mycompany.eventoshyrule
      > com.mycompany.eventoshyrule.c
        > ActividadControlador.java
        > AsistenciaControlador.java
        > CertificadoControlador.java
        > EventoControlador.java
        > InscripcionControlador.java
        > PagoControlador.java
        > ParticipanteControlador.java
        > ReporteControlador.java
      > com.mycompany.eventoshyrule.c
        > ActividadDAO.java
        > AsistenciaDAO.java
        > CertificadoDAO.java
        > EventoDAO.java
        > InscripcionDAO.java
        > PagoDAO.java
        > ParticipanteDAO.java
        > actividad.java
        > Asistencia.java
        > Certificado.java
        > DBConnection.java
        > Evento.java
        > Inscripcion.java
        > Pago.java
        > Participante.java
      > com.mycompany.eventoshyrule.c
        > Parser.java
      > com.mycompany.eventoshyrule.v
        > FrmActividad.java
        > FrmAsistencia.java
        > FrmCertificado.java
        > FrmEvento.java
        > FrmInscripcion.java
        > FrmListaActividades.java
        > FrmListaEventos.java
        > FrmListarParticipantes.java
        > FrmPago.java
        > FrmParticipante.java
Source History ...va ActividadControlador.java AsistenciaControlador.java CertificadoControlador.java Parser.java MainFrame.java pom.xml [Practica01] FrmParticipante.java FrmEvento.java
13 */
14 public class AsistenciaControlador {
15     private AsistenciaDAO asistenciaDAO;
16     private ParticipanteDAO participanteDAO;
17     private ActividadDAO actividadDAO;
18
19     public AsistenciaControlador(){
20         this.asistenciaDAO = new AsistenciaDAO();
21         this.participanteDAO = new ParticipanteDAO();
22         this.actividadDAO = new ActividadDAO();
23     }
24
25     public boolean registrarAsistencia(String correoParticipante, String codigoActividad){
26         //validar que exista el participante
27         Participante participante = participanteDAO.buscarPorCorreo(correoParticipante);
28         if (participante == null) {
29             System.out.println("No existe un participante con el correo: " + correoParticipante);
30             return false;
31         }
32
33         //validar actividad
34         Actividad actividad = actividadDAO.buscarPorCodigo(codigoActividad);
35         if (actividad == null) {
36             System.out.println("No existe actividad con codigo: " + codigoActividad);
37             return false;
38         }
39
40         //registrar asistencia
41         Asistencia asistencia = new Asistencia(0, correoParticipante, codigoActividad);
42         return asistenciaDAO.insertar(asistencia);
43     }
44
45     //listar asistencias
46     public List<Asistencia> listarAsistencias(){
47         return asistenciaDAO.listarTodos();
48     }
49
50     //busqueda por id
51     public Asistencia buscarAsistencia(int idAsistencia){
52         return asistenciaDAO.buscarPorId(idAsistencia);
53     }
54
55     //eliminar
56     public boolean eliminarAsistencia(int idAsistencia){
57         return asistenciaDAO.eliminar(idAsistencia);
58     }
59 }
60

```

42:16 INS Unix (LF)

Clase EventoControlador: Gestiona la creacion y administracion de eventos. Implementa validaciones exhaustivas que aseguran que cada evento creado cumpla con todos los requisitos.

EventosHyrule - Apache NetBeans IDE 20

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
...va EventoControlador.java IncripcionControlador.java ParticipanteControlador.java PagoControlador.java ActividadControlador.java AsistenciaControlador.java CertificadoC...
Source History ...va EventoControlador.java IncripcionControlador.java ParticipanteControlador.java PagoControlador.java ActividadControlador.java AsistenciaControlador.java CertificadoC...
367.61332MB Search (Ctrl+F)
Projects x Files Services
> CalculadoraConPlas
> ejercicioNumerosAleatorios
> EventosHyrule [app-funcional-1]
  > Source Packages
    > com.mycompany.eventoshyrule
      > com.mycompany.eventoshyrule.c
        > ActividadControlador.java
        > AsistenciaControlador.java
        > CertificadoControlador.java
        > EventoControlador.java
        > IncripcionControlador.java
        > PagoControlador.java
        > ParticipanteControlador.java
        > ReporteControlador.java
      > com.mycompany.eventoshyrule.c
        > ActividadDAO.java
        > AsistenciaDAO.java
        > CertificadoDAO.java
        > DBConnection.java
        > EventoDAO.java
        > IncripcionDAO.java
        > PagoDAO.java
        > ParticipanteDAO.java
        > actividad.java
        > Asistencia.java
        > Certificado.java
        > DBConnection.java
        > Evento.java
        > Incripcion.java
        > Pago.java
        > Participante.java
      > com.mycompany.eventoshyrule.c
        > Parser.java
      > com.mycompany.eventoshyrule.v
        > FrmActividad.java
        > FrmAsistencia.java
        > FrmCertificado.java
        > FrmEvento.java
        > FrmInscripcion.java
        > FrmListaActividades.java
        > FrmListaEventos.java
        > FrmListarParticipantes.java
        > FrmPago.java
        > FrmParticipante.java
Source History ...va EventoControlador.java IncripcionControlador.java ParticipanteControlador.java PagoControlador.java ActividadControlador.java AsistenciaControlador.java CertificadoC...
19
20     public boolean crearEvento(String codigo, Date fecha, String tipoEvento, String titulo, String ubicacion, int cupoMaximo, BigDecimal costoIncripcion) {
21         if (codigo==null || codigo.isEmpty()) {
22             System.out.println("El codigo del evento no puede estar vacio");
23             return false;
24         }
25
26         if (fecha==null) {
27             System.out.println("La fecha del evento no puede estar vacia");
28             return false;
29         }
30
31         if (tipoEvento==null || tipoEvento.isEmpty()) {
32             System.out.println("Debe indicarse el tipo de evento");
33             return false;
34         }
35
36         if (titulo==null || titulo.isEmpty()) {
37             System.out.println("El titulo del evento no puede estar vacio");
38             return false;
39         }
40
41         if (cupoMaximo<=0) {
42             System.out.println("El cupo maximo del evento debe ser mayor que cero");
43             return false;
44         }
45
46         if (costoIncripcion.compareTo(BigDecimal.ZERO)< 0) {
47             System.out.println("El costo de la inscripcion al evento no puede ser menor a cero");
48             return false;
49         }
50
51         Evento evento = new Evento(codigo, fecha, tipoEvento, titulo, ubicacion, cupoMaximo, costoIncripcion);
52         return eventoDAO.insertar(evento);
53     }
54
55     //listar a todos los eventos
56     public List<Evento> listarEventos(){
57         return eventoDAO.listarTodos();
58     }
59
60     //buscar un evento por codigo
61     public Evento buscarEvento(String codigo){
62         if (codigo==null || codigo.isEmpty()) {
63             System.out.println("Debe ingresar un codigo para buscar");
64             return null;
65         }
66     }

```

78:5 INS Unix (LF)

Clase IncripcionControlador: Gestiona el proceso completo de inscripcion de participantes a eventos. Evita duplicados.

EventoHyrule - Apache NetBeans IDE 20

```
File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config...> Search (Ctrl+)
```

Projects x Files Services

Source History

```
24
25
26     // crear una inscripcion con validaciones
27     public boolean crearInscripcion (String correoParticipante, String codigoEvento, String tipoIncripcion){
28         //validar que exista el participante
29         Particinante participante = participanteDAO.buscarPorCorreo(correoParticipante);
30         if (participante == null) {
31             System.out.println("No existe un participante con el correo: " + correoParticipante);
32             return false;
33         }
34
35         //validar que exista el evento
36         Evento evento = eventoDAO.buscarPorCodigo(codigoEvento);
37         if (evento == null) {
38             System.out.println("No existe un evento con el codigo: " + codigoEvento);
39             return false;
40         }
41
42         //validar que no este ya inscrito
43         List<Inscripcion> inscripciones = inscripcionDAO.listarTodos();
44         for (Inscripcion ins : inscripciones) {
45             if (ins.getCorreoParticipante().equals(correoParticipante) && ins.getCodigoEvento().equals(codigoEvento)) {
46                 System.out.println("El participante ya esta inscrito en el evento");
47                 return false;
48             }
49
50         }
51
52         //Crear la inscripcion al cumplir validaciones
53         Incripcion inscripcion = new Incripcion(0, correoParticipante, codigoEvento, tipoIncripcion, false);
54         return inscripcionDAO.insertar(inscripcion);
55
56
57         // listar todas las inscripciones
58         public List<Inscripcion> listarInscripciones(){
59             return inscripcionDAO.listarTodos();
60         }
61
62         //validar inscripcion (tras pago)
63         public boolean validarInscripcion (int idInscripcion){
64             Incripcion inscripcion = inscripcionDAO.buscarPorId(idInscripcion);
65             if (inscripcion == null) {
66                 System.out.println("No existe una inscripcion con el ID: " + idInscripcion);
67                 return false;
68             }
69             inscripcion.setValidada(true);
70             return inscripcionDAO.actualizar(inscripcion);
71         }
72
73     }
74 }
```

75:35 INS Unix (LF)

Clase PagoControlador: Administra el proceso de pagos y su relacion con las validaciones de inscripcion.

EventoHyrule - Apache NetBeans IDE 20

```
File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config...> Search (Ctrl+)
```

Projects x Files Services

Source History

```
28
29
30
31     // realizar el pago (y validar la inscripcion)
32     public boolean registrarPago(String correoParticipante, String codigoEvento, String metodoPago, BigDecimal monto){
33         //verificar que existe el participante
34         Particinante participante = participanteDAO.buscarPorCorreo(correoParticipante);
35         if (participante==null) {
36             System.out.println("No existe un participante con el correo: " + correoParticipante);
37             return false;
38         }
39
40         // validar existencia de evento
41         Evento evento = eventoDAO.buscarPorCodigo(codigoEvento);
42         if (evento==null) {
43             System.out.println("No existe un evento con el codigo: " + codigoEvento);
44             return false;
45         }
46
47         // validar existencia de inscripcion previa
48         Incripcion inscripcion = null;
49         //busqueda manual en memoria con un bucle
50         for (Inscripcion i : inscripcionDAO.listarTodos()) {
51             if (i.getCorreoParticipante().equals(correoParticipante) && i.getCodigoEvento().equals(codigoEvento)) {
52                 inscripcion = i;
53                 break;
54             }
55         }
56         if (inscripcion==null) {
57             System.out.println("No existe un participante con el correo: " + correoParticipante);
58             return false;
59         }
60
61         //registrar el pago
62         Pago pago = new Pago(0, correoParticipante, codigoEvento, metodoPago, monto);
63         return pagoDAO.insertar(pago);
64
65         //validar inscripcion
66         public boolean validarInscripcion (String correoParticipante, String codigoEvento){
67             Incripcion inscripcion = null;
68             for (Inscripcion i : inscripcionDAO.listarTodos()) {
69                 if (i.getCorreoParticipante().equals(correoParticipante) && i.getCodigoEvento().equals(codigoEvento)) {
70                     inscripcion = i;
71                     break;
72                 }
73             }
74             if (inscripcion == null) {
75 }
```

83:10 INS Unix (LF)

EventosHyrule - Apache NetBeans IDE 20

```
File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config> Search (Ctrl+F)
```

Projects x Files Services

Source History

```
64 //validar inscripcion
65 public boolean validarIncripcion (String correoParticipante, String codigoEvento){
66     Incripcion inscripcion = null;
67     for (Incripcion i : inscripcionDAO.listarTodos()) {
68         if (i.getCorreoParticipante().equals(correoParticipante) && i.getCodigoEvento().equals(codigoEvento)) {
69             inscripcion = i;
70             break;
71         }
72     }
73     if (inscripcion == null) {
74         System.out.println("No existe una inscripcion para validar");
75         return false;
76     }
77     if (inscripcion.isValidada()) {
78         System.out.println("La inscripcion ya fue validada");
79         return false;
80     }
81     inscripcion.setValidada(true);
82     return inscripcionDAO.actualizar(inscripcion);
83 }
84
85 //listar todos los pagos
86 public List<Pago> listarPagos(){
87     return pagoDAO.listarTodos();
88 }
89
90 //buscar pago por id
91 public Pago buscarPago(int idPago){
92     return pagoDAO.buscarPorId(idPago);
93 }
94
95 //eliminar pago (opcional ante cualquier error)
96 public boolean eliminarPago(int idPago){
97     if (pagoDAO.buscarPorId(idPago)==null) {
98         System.out.println("No se encontro el pago o no existe");
99         return false;
100    }
101    return pagoDAO.eliminar(idPago);
102 }
```

83:10 INS Unix (LF)

Clase ParticipanteControlador: Responsable de la gestión de usuarios/participantes del sistema.

EventosHyrule - Apache NetBeans IDE 20

```
File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help <default config> Search (Ctrl+F)
```

Projects x Files Services

Source History

```
15 }
16
17 //crear participante
18 public boolean crearParticipante(String nombreCompleto, String tipoParticipante, String institucion, String correo){
19     if (correo==null || correo.isEmpty()) {
20         System.out.println("El correo del participante no puede estar vacio");
21         return false;
22     }
23     if (nombreCompleto==null || nombreCompleto.isEmpty()) {
24         System.out.println("El nombre del participante no puede estar vacio");
25         return false;
26     }
27     if (tipoParticipante==null || tipoParticipante.isEmpty()) {
28         System.out.println("Debe indicar el tipo de participante");
29         return false;
30     }
31     Particinante participante = new Particinante (nombreCompleto, tipoParticipante, institucion, correo);
32     return participanteDAO.insertar(participante);
33 }
34
35 //listar a los participantes
36 public List<Particinante> listarParticipantes(){
37     return participanteDAO.listarTodos();
38 }
39
40 //buscar a un participante por correo
41 public Particinante buscarParticipante(String correo){
42     if (correo==null || correo.isEmpty()) {
43         System.out.println("Debe ingresar un correo para buscar");
44         return null;
45     }
46     return participanteDAO.buscarPorCorreo(correo);
47 }
48
49 //actualizacion de participante
50 public boolean actualizarParticipante(String nombreCompleto, String tipoParticipante, String institucion, String correo){
51     if (buscarParticipante(correo)==null) {
52         System.out.println("No se puede actualizar o el participante no esta registrado");
53         return false;
54     }
55     Particinante participante = new Particinante (correo, nombreCompleto, tipoParticipante, institucion);
56     return participanteDAO.actualizar(participante);
57 }
58
59 //eliminar a un participante
60 }
```

39:53 INS Unix (LF)

Clase ReporteControlador: Especializado en la generación de reportes y estadísticas del sistema. Proporciona funcionalidades de reporting y análisis de datos.

EventosHyrule - Apache NetBeans IDE 20

File Ed View Navigate Source Refactor Run Debug Profile Team Tools Window Help 186.81.128.161 Search (Ctrl+)

Projects x Files Services ...va FrmAsistencia.java x FrmCertificado.java x FrmListaParticipantes.java x FrmListaEventos.java x FrmListaActividades.java x ReporteControlador.java x FrmReportes.java x

Source History

```
private ActividadDAO actividadDAO;
private AsistenciaDAO asistenciaDAO;
private IncripcionDAO inscripcionDAO;
```

```
public ReporteControlador(){
    this.participanteDAO = new ParticipanteDAO();
    this.eventoDAO = new EventoDAO();
    this.actividadDAO = new ActividadDAO();
    this.asistenciaDAO = new AsistenciaDAO();
    this.inscripcionDAO = new IncripcionDAO();
}
```

```
try {
    Files.createDirectories(Paths.get(RUTA_REPORTES, "participantes"));
    Files.createDirectories(Paths.get(RUTA_REPORTES, "actividades"));
    Files.createDirectories(Paths.get(RUTA_REPORTES, "eventos"));
}
```

```
} catch (IOException e) {
    System.out.println("Error creando carpetas de reportes: " + e.getMessage());
}
```

```
public void generarReporteParticipantes(String codigoEvento, String tipoParticipante, String institucion){
    String nombreArchivo = RUTA_REPORTES + "/participantes/reporte_participantes_" + codigoEvento + ".html";
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(nombreArchivo))){
        writer.write("<html><head><title>Reporte de Participantes</title></head><body>");
        writer.write("<h1>Reporte de participantes del evento " + codigoEvento + "</h1>");

        writer.write("</body></html>");
        System.out.println("Reporte generado: " + nombreArchivo);
    } catch (IOException e) {
        System.out.println("Error al generar reporte" + e.getMessage());
    }
}
```

```
public void generarReporteActividades(String codigoEvento, String tipoActividad, String correoEncargado){
    String nombreArchivo = RUTA_REPORTES + "/actividades/reporte_actividades_" + codigoEvento + ".html";
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(nombreArchivo))){
        writer.write("<html><head><title>Reporte de Actividades</title></head><body>");
        writer.write("<h1>Reporte de actividades del evento " + codigoEvento + "</h1>");

        writer.write("</body></html>");
        System.out.println("Reporte generado: " + nombreArchivo);
    } catch (IOException e) {
        System.out.println("Error al generar reporte" + e.getMessage());
    }
}
```

66:34 INS Unix (LF)



DIAGRAMA DE ENTIDAD-RELACION

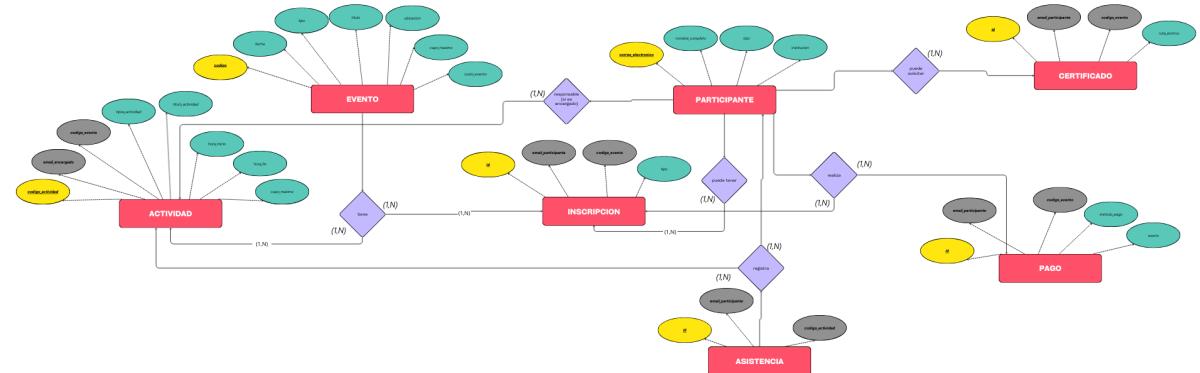


DIAGRAMA DE TABLAS

