Manual Técnico - Sistema de Gestión de Eventos

1. Introducción

1.1. Propósito

Este manual técnico documenta la arquitectura, diseño e implementación del Sistema de Gestión de Eventos desarrollado para la empresa Triforce Software, destinado a la administración de eventos en el Reino de Hyrule.

1.2. Alcance

El sistema permite el registro y gestión de eventos, participantes, actividades, inscripciones, pagos, asistencias y generación de reportes en formato HTML.

2. Arquitectura del Sistema

2.1. Diagrama de Arquitectura

Capa de Presentación (Swing) → Capa de Controlador → Capa de Modelo/DAO → Base de Datos

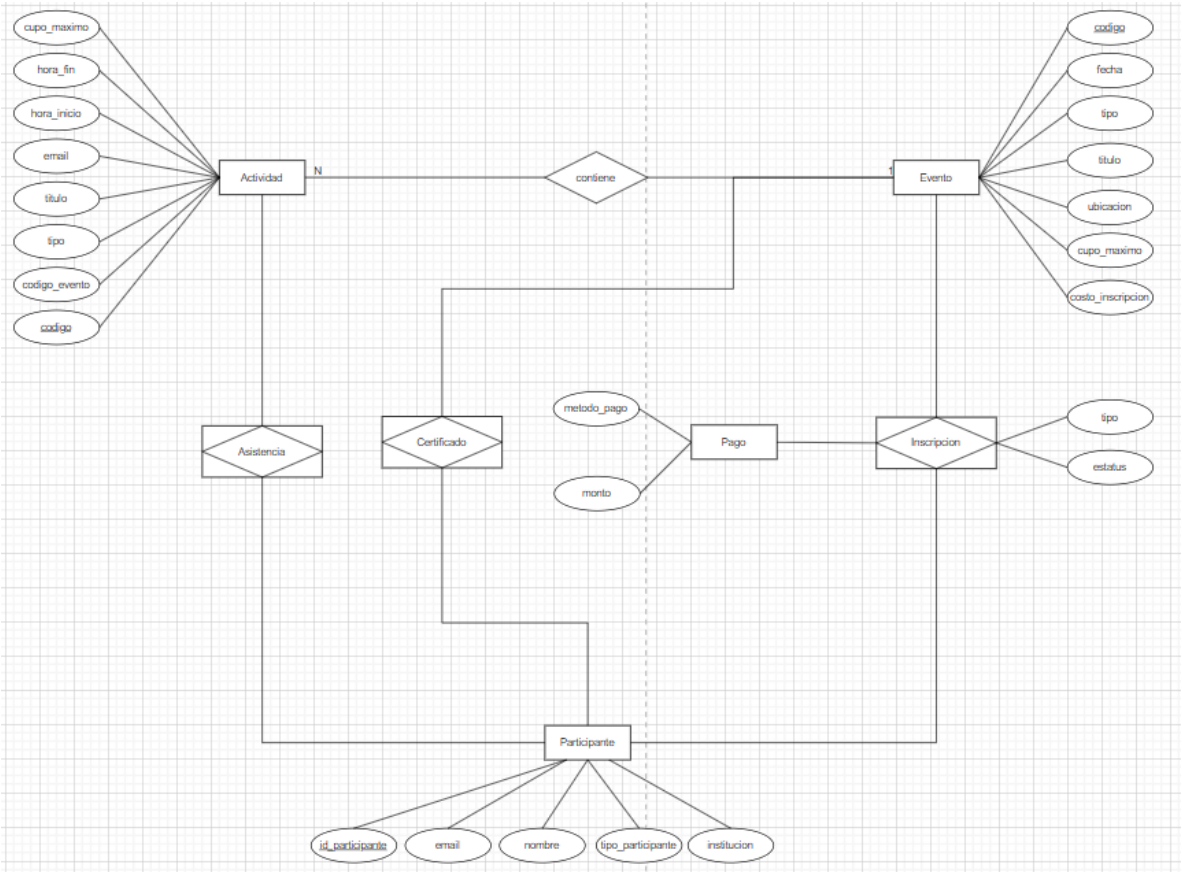
2.2. Tecnologías Utilizadas

- **Lenguaje**: Java 8+
- **Interfaz gráfica**: Java Swing

- **Base de datos**: MySQL 8.0+
- **Control de versiones**: Git
- **Patrones de diseño**: MVC, DAO, Singleton

3. Diagrama Entidad-Relación (E/R)

3.1. Diagrama Conceptual



3.2. Diagrama Físico de Base de Datos

```
-- -----------------------------------------------------
-- Table `practica1`.`participante`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `practica1`.`participante` (
  `id` INT NOT NULL,
  `email` VARCHAR(150) NOT NULL,
  `nombre` VARCHAR(45) NULL,
  `tipo` VARCHAR(45) NULL,
  `institucion_procedencia` VARCHAR(150) NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE);
-- -----------------------------------------------------
-- Table `practica1`.`evento`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `practica1`.`evento` (
  `codigo` VARCHAR(45) NOT NULL,
  `fecha` DATE NULL,
  `tipo` VARCHAR(45) NULL,
  `titulo_evento` VARCHAR(200) NULL,
  `ubicacion` VARCHAR(150) NULL,
  `cupo_maximo` INT NULL,
  `costo_inscripcion` DECIMAL(10,2) NULL,
  PRIMARY KEY (`codigo`));
-- -----------------------------------------------------
-- Table `practica1`.`actividad`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `practica1`.`actividad` (
  `codigo` VARCHAR(45) NOT NULL,
  `tipo` VARCHAR(45) NULL,
  `titulo` VARCHAR(200) NULL,
  `hora_inicio` TIME NULL,
  `hora_fin` TIME NULL,
  `cupo_maximo` INT NULL,
  `id_participante_encargado` INT NOT NULL,
  `evento_codigo_evento` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`codigo`),
  CONSTRAINT `fk_actividad_id_encargado`
    FOREIGN KEY (`id_participante_encargado`)
    REFERENCES `practica1`.`participante` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_actividad_codigo_evento`
    FOREIGN KEY (`evento_codigo_evento`)
    REFERENCES `practica1`.`evento` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
-- -----------------------------------------------------
-- Table `practica1`.`pago`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `practica1`.`pago` (
  `id_pago` INT NOT NULL,
  `tipo_pago` VARCHAR(45) NULL,
  `monto` DECIMAL(10,2) NULL,
  `evento_codigo` VARCHAR(45) NOT NULL,
```

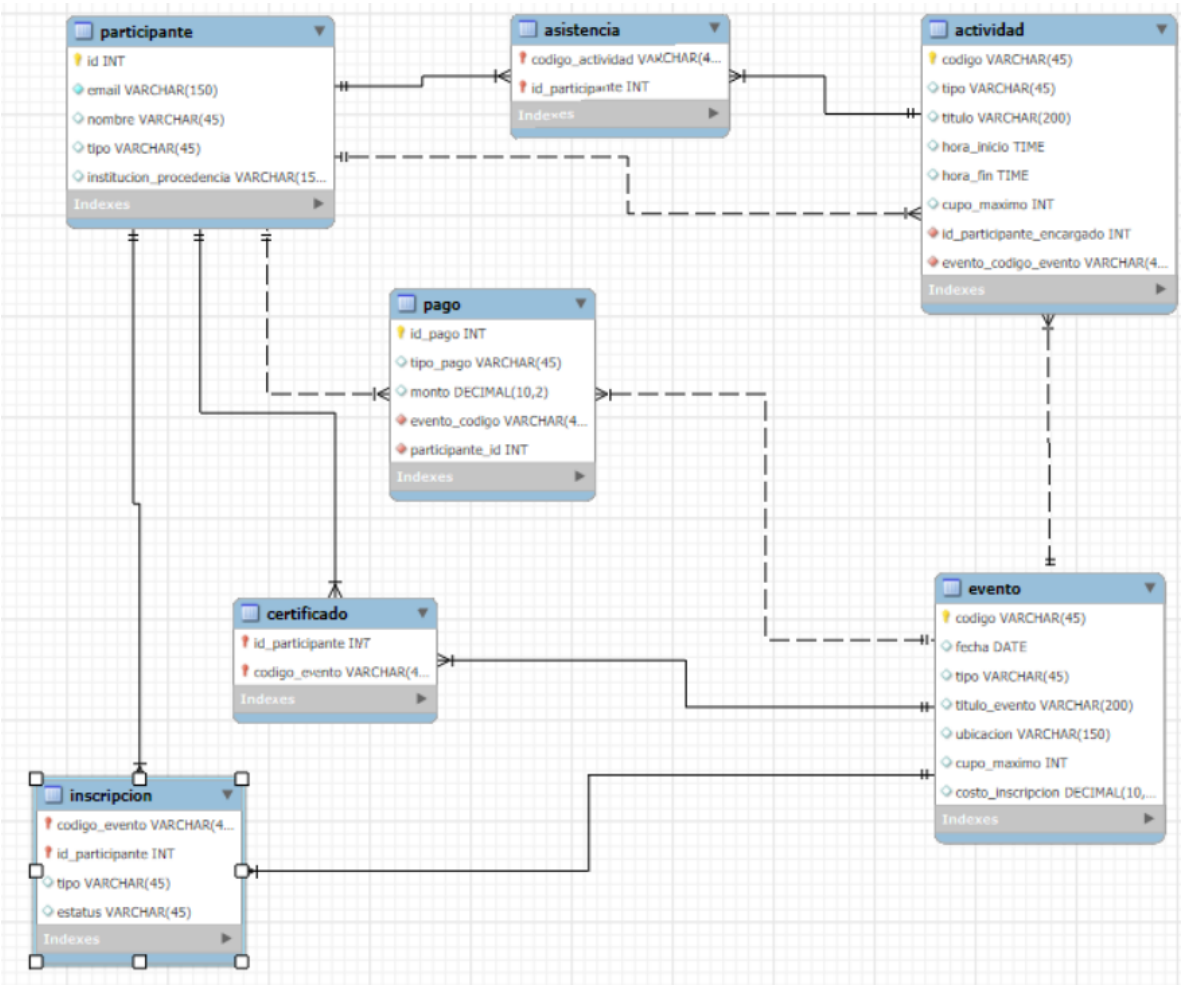
```
  `participante_id` INT NOT NULL,
  PRIMARY KEY (`id_pago`),
  CONSTRAINT `fk_pago_codigo_evento`
    FOREIGN KEY (`evento_codigo`)
    REFERENCES `practica1`.`evento` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pago_id_participante`
    FOREIGN KEY (`participante_id`)
    REFERENCES `practica1`.`participante` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
-- -----------------------------------------------------
-- Table `practica1`.`asistencia`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `practica1`.`asistencia` (
  `codigo_actividad` VARCHAR(45) NOT NULL,
  `id_participante` INT NOT NULL,
  PRIMARY KEY (`codigo_actividad`, `id_participante`),
  CONSTRAINT `fk_asistencia_codigo_actividad`
    FOREIGN KEY (`codigo_actividad`)
    REFERENCES `practica1`.`actividad` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_asistencia_id_participante`
    FOREIGN KEY (`id_participante`)
    REFERENCES `practica1`.`participante` (`id`)
    ON DELETE NO ACTION
```

```
  ON UPDATE NO ACTION);

-- -----------------------------------------------------
-- Table `practica1`.`inscripcion`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `practica1`.`inscripcion` (
  `codigo_evento` VARCHAR(45) NOT NULL,
  `id_participante` INT NOT NULL,
  `tipo` VARCHAR(45) NULL,
  `estatus` VARCHAR(45) NULL,
  PRIMARY KEY (`codigo_evento`, `id_participante`),
  CONSTRAINT `fk_inscripcion_codigo_evento`
    FOREIGN KEY (`codigo_evento`)
    REFERENCES `practica1`.`evento` (`codigo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_inscripcion_id_participante`
    FOREIGN KEY (`id_participante`)
    REFERENCES `practica1`.`participante` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

-- -----------------------------------------------------
-- Table `practica1`.`certificado`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `practica1`.`certificado` (
  `id_participante` INT NOT NULL,
  `codigo_evento` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id_participante`, `codigo_evento`),
  CONSTRAINT `fk_certificado_id_participante`
```

```
 FOREIGN KEY (`id_participante`)
 REFERENCES `practica1`.`participante` (`id`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
CONSTRAINT `fk_certificado_codigo_evento`
 FOREIGN KEY (`codigo_evento`)
 REFERENCES `practica1`.`evento` (`codigo`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION);
```



4. Diagrama de Clases

4.1. Clases del Modelo

```
// Entidades principales
public class Evento {
   private String codigo;
   private LocalDate fecha;
   private TipoEvento tipo;
   private String titulo;
   private String ubicacion;
   private int cupoMaximo;
   private double costoInscripcion;
   // getters, setters, constructores
}

public class Participante  {
   private String nombre;
   private TipoParticipante tipo;
   private String institucion;
   private String email;
   private TipoEstatus estatus;
   // getters, setters, constructores
}

public class Actividad  {
   private String codigo;
   private TipoActividad tipo;
```

```
    private String titulo;
    private LocalTime horaInicio;
    private LocalTime horaFin;
    private int cupoMaximo;
    private int idEncargado;
    private String codigoEvento;
    // getters, setters, constructores
}

public class Inscripcion {
    private int idParticipante;
    private String codigoEvento;
    private TipoInscripcion tipo;
    private TipoEstatus estatus;
    // getters, setters, constructores
}

public class Pago {
    private int idParticipante;
    private String codigoEvento;
    private MetodoPago metodo;
    private double monto;
    private LocalDate fechaPago;
    private TipoEstatus estatus;
    // getters, setters, constructores
}
```

```
public class Asistencia  {
   private int idParticipante;
   private String codigoActividad;
   private LocalDateTime fechaAsistencia;
   // getters, setters, constructores
}

public class Certificado {
   private int idParticipante;
   private String codigoEvento;
   private String codigoCertificado;
   private LocalDate fechaEmision;
   private String rutaArchivo;
   // getters, setters, constructores
}
```

4.2. Enumerados

java

```
public enum TipoEvento {
   CHARLA, CONGRESO, TALLER, DEBATE
}

public enum TipoParticipante {
   ESTUDIANTE, PROFESIONAL, INVITADO
}
```

```
public enum TipoActividad {
   CHARLA, TALLER, DEBATE, OTRA
}

public enum TipoInscripcion {
   ASISTENTE, CONFERENCISTA, TALLERISTA, OTRO
}

public enum TipoEstatus {
   PENDIENTE, CONFIRMADO, CANCELADO, RECHAZADO
}

public enum MetodoPago {
   EFECTIVO, TRANSFERENCIA, TARJETA
}
```

5. Componentes de la Interfaz Gráfica

5.1. Estructura de Ventanas

```
MainFrame (JFrame)
 ├── JDesktopPane
 │    ├── LogWindow (JInternalFrame)
 │    ├── EventoFormInternalFrame
 │    ├── ParticipanteFormInternalFrame
 │    ├── EventoListInternalFrame
 │    ├── ParticipanteListInternalFrame
```

```
│   ├── ReporteParticipantesInternalFrame
│   ├── ReporteActividadesInternalFrame
│   └── ReporteEventosInternalFrame
└── JMenuBar
    ├── Archivo Menu
    ├── Eventos Menu
    ├── Participantes Menu
    └── Reportes Menu
```

6. Procesamiento de Archivos

6.1. Esquema de Procesamiento

java

```
public class FileProcessor {
    public void procesarArchivo(File archivo) {
        try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                procesarLinea(linea.trim());
            }
        }
    }

    private String procesarLinea(String linea) {
        return switch (extraerComando(linea)) {
            case "REGISTRO_EVENTO" -> procesarRegistroEvento(linea);
            case "REGISTRO_PARTICIPANTE" -> procesarRegistroParticipante(linea);
            // ... otros comandos
```

```
        default -> throw new IllegalArgumentException("Instrucción no reconocida");
      };
  }
}
```

6.2. Formatos de Instrucciones

```
text
REGISTRO_EVENTO("EVT-001","25/08/2025","CHARLA","Título","Ubicación",150,50.00)
REGISTRO_PARTICIPANTE("Nombre","ESTUDIANTE","Institución","email@ejemplo.com
")
INSCRIPCION("email@ejemplo.com","EVT-001","ASISTENTE")
PAGO("email@ejemplo.com","EVT-001","EFECTIVO",50.00)
```

7. Validaciones

7.1. Clase Validador

```
java
public class Validador {
  // Validaciones genéricas
  public static void validarNoVacio(String valor, String campo)
  public static void validarLongitud(String valor, int min, int max, String campo)
  public static void validarEmail(String email)

  // Validaciones específicas
  public static void validarCodigoEvento(String codigo)
  public static void validarCodigoActividad(String codigo)
  public static void validarFecha(String fechaStr, String formato)
  public static void validarHora(String horaStr)
```

}

8. Reportes HTML

8.1 Estructura de Reportes

```
public class GeneradorReportes {
   public void generarReporteHTML(String titulo, List<?> datos, String rutaSalida) {
      // Plantilla HTML con CSS incorporado
      // Tablas dinámicas con datos
      // Estilos responsivos
   }
}
```

8.2. Tipos de Reportes

1. **Reporte de Participantes**: Por evento con filtros opcionales
2. **Reporte de Actividades**: Con estadísticas de asistencia
3. **Reporte de Eventos**: Con información financiera y de participación

9. Configuración e Instalación

9.1. Requisitos del Sistema

- Java JDK 8 o superior
- MySQL 8.0+
- 4GB RAM mínimo
- 500MB espacio en disco

9.2. Scripts de Base de Datos

```
bash
# Ejecutar script de creación
mysql -u usuario -p < database/schema.sql
mysql -u usuario -p < database/data.sql
```

11.3. Estructura de Proyecto

text

```
src/
 ├── main/
 │   ├── java/
 │   │   ├── controllers/
 │   │   ├── models/
 │   │   ├── daos/
 │   │   ├── gui/
 │   │   ├── utils/
 │   │   └── Main.java
 │   └── resources/
 ├── database/
 │   ├── schema.sql
 │   └── data.sql
 └── docs/
     ├── manual_tecnico.md
     └── manual_usuario.md
```

Este manual técnico proporciona una visión completa de la arquitectura y implementación del sistema, cumpliendo con todos los requisitos especificados en la práctica.