Informe de Desarrollo de Aplicación: Concurso de Ajedrez.

> El objetivo de este proyecto es desarrollar una aplicación que permita al usuario gestionar torneos de ajedrez guardando la información en una base de datos relacional.

Requisitos

El usuario debe poder **consultar** las diferentes rondas del torneo junto con los participantes, horarios y tiempo de partida. Podrá consultar por eliminatorias los diferentes participantes.

Funcionalidad

Este proyecto implementa una ***base de datos relacional*** implementada en un sistema gestor de bases de datos para gestionar el torneo y sus datos. Debido a los requerimientos del cliente, necesitamos montar el archivo .sql dentro de un SGBD y para poder tratar los datos utilizaremos Java junto a JBDC.

El SGBD utilizado en este caso es MySQL con MySQL Workbench para consultar los datos.

Análisis

Diagrama de pantallas

En este caso solo tendremos una ventana para poder consultar todos los datos. Una interfaz más que suficiente para poder consultar los diferentes encuentros de ajedrez.

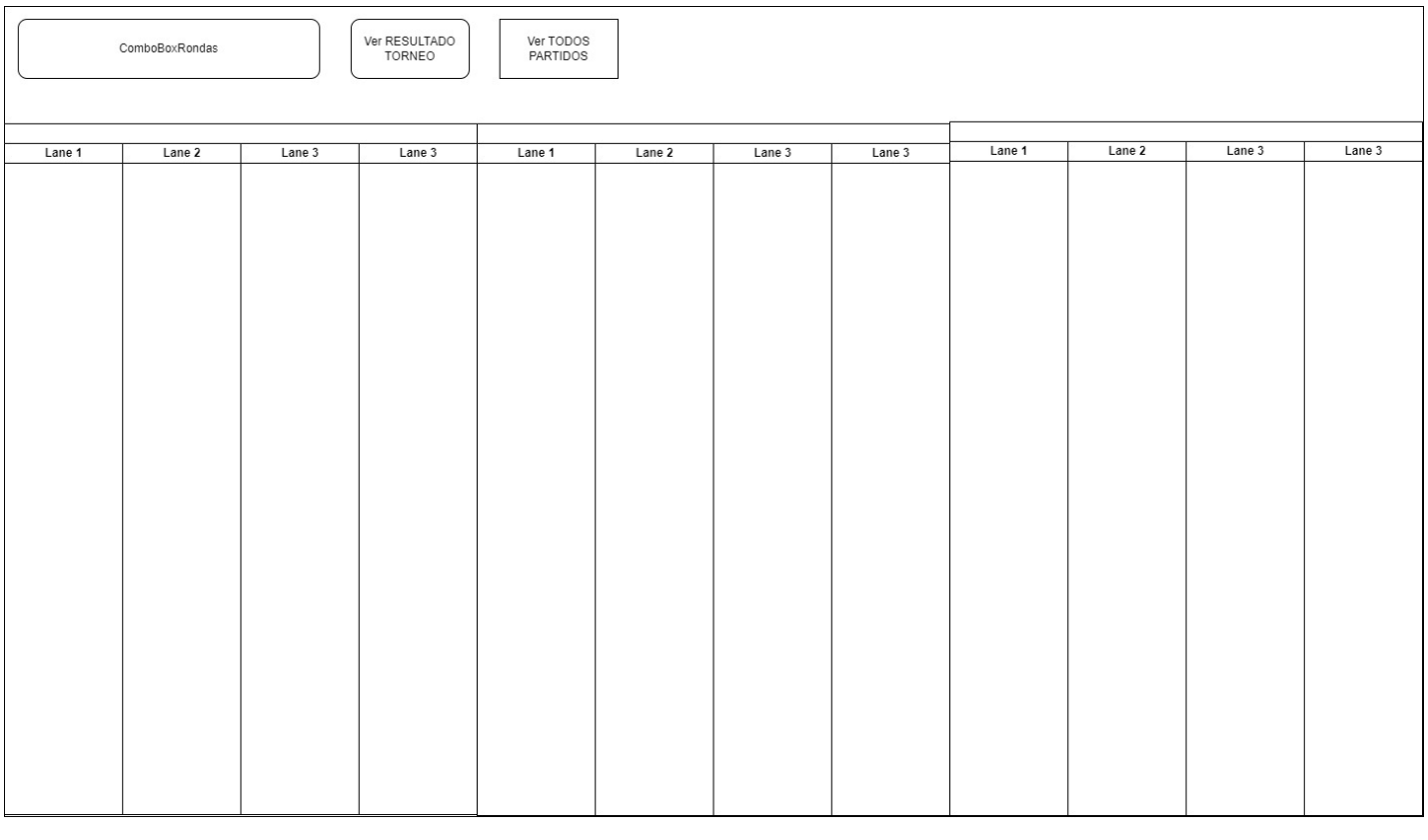


Diagrama Entidad-Relacion

En este diagrama podemos observar las cuatro entidades que van a componer nuestra base de datos.

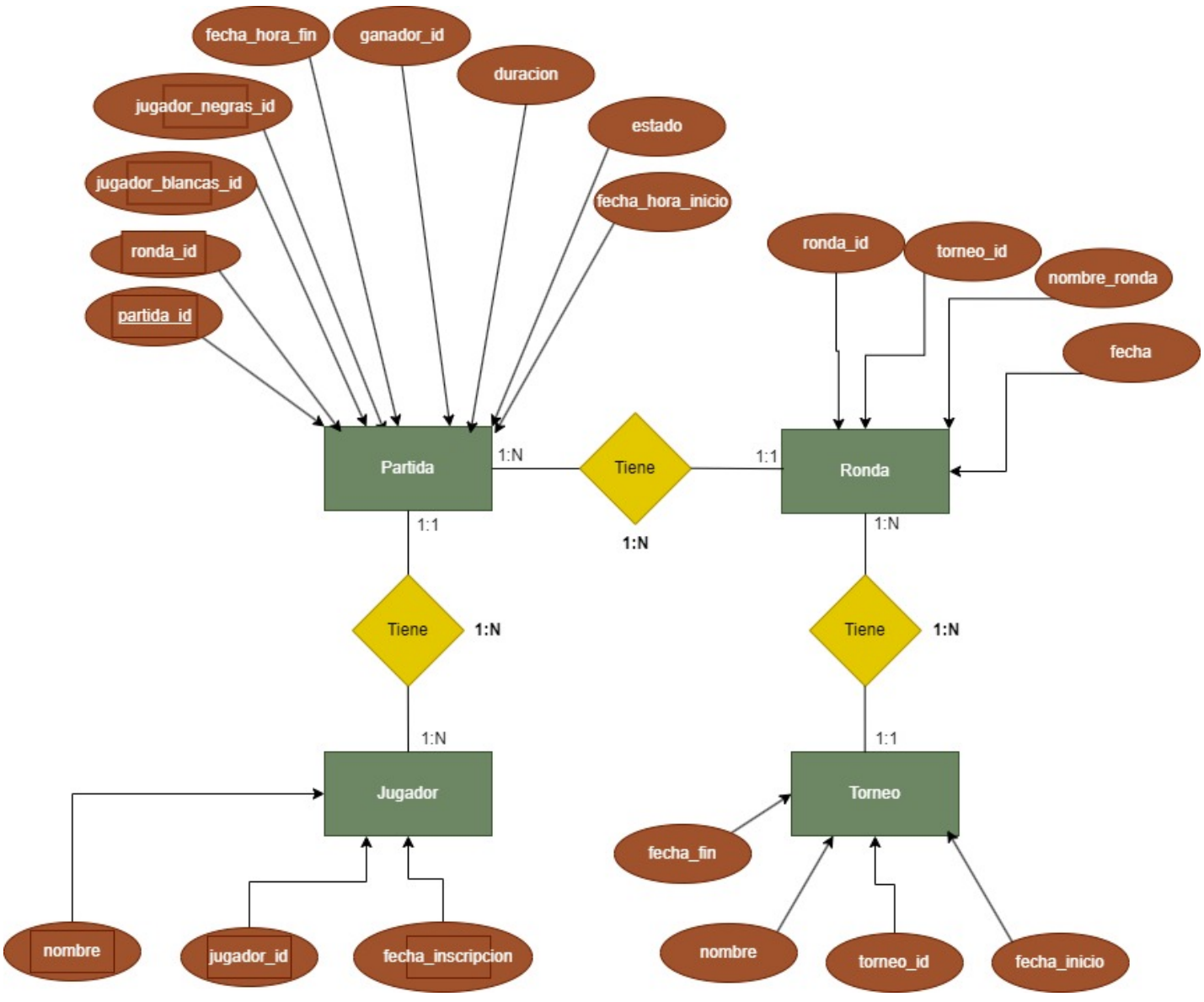


Diagrama Relacional

El diagrama relacional nos da la capacidad de ver mas visualmente las relaciones foráneas entre tablas.

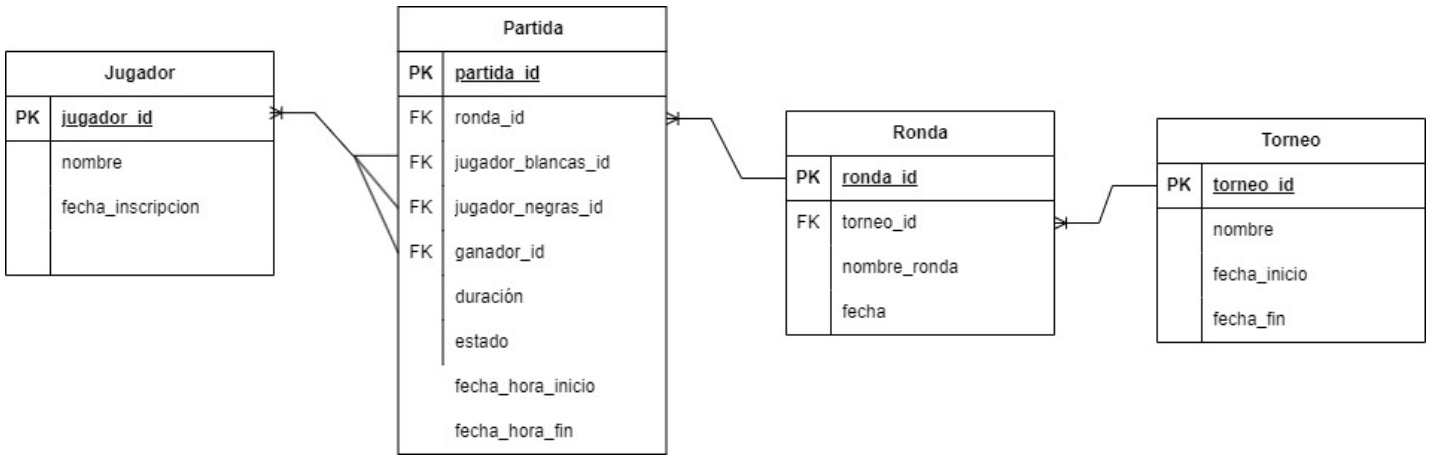
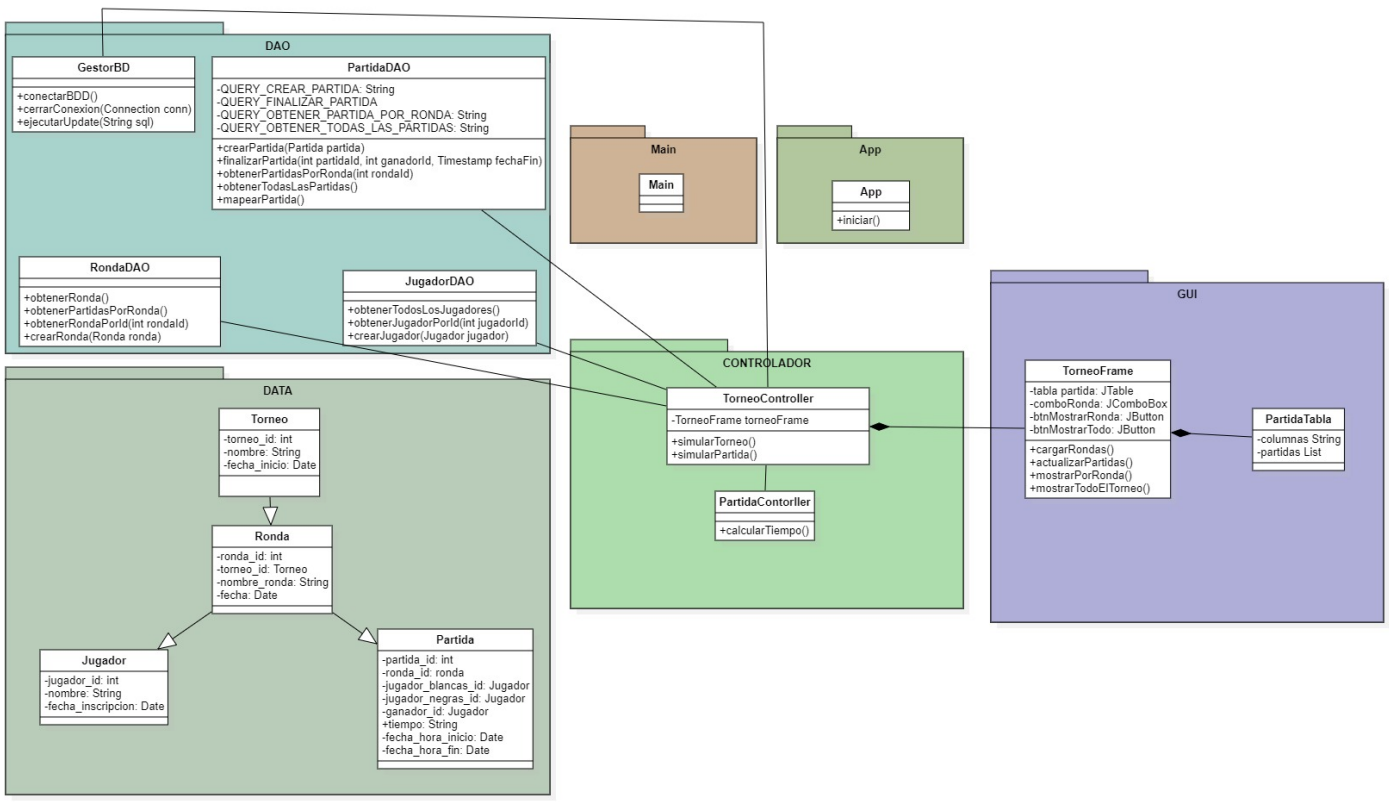


Diagrama de Clases

Aquí podemos observar que se ha encapsulado cada paquete para que cada uno trate la información correspondiente. El paquete ***DAO*** encapsula la lógica de las consultas SQL y obtiene la información de las tablas de las bases de datos. Después, tenemos el paquete ***Controlador***, el cual nos permite tratar los datos utilizados en ***DAO*** y pasarlos al paquete ***GUI***.

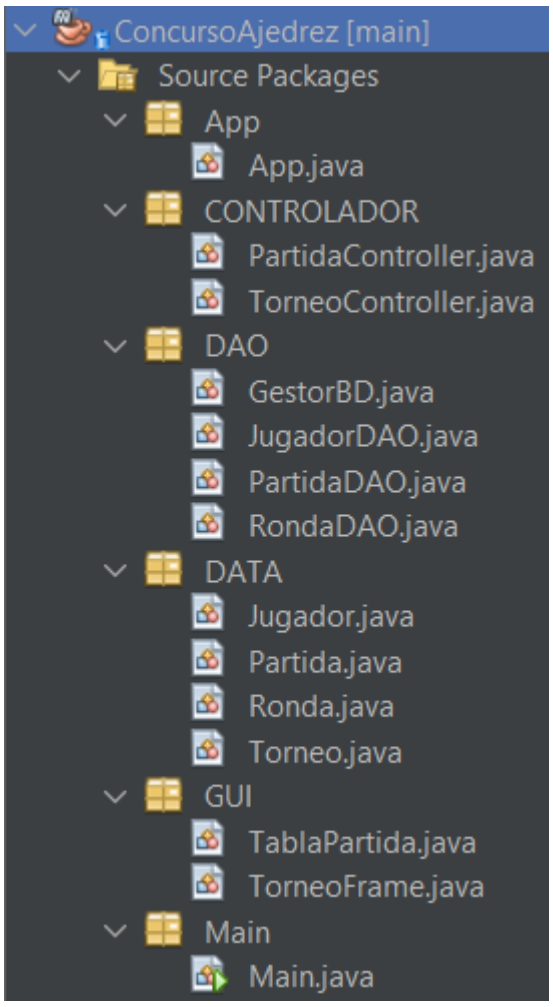
Básicamente, se ha tratado de seguir el principio de ***separación de responsabilidades*** tratando de hacer un Modelo-Vista-Controlador**(MVC)**.



Código

En este apartado, se muestran las funciones mas destacadas de la aplicación, incluyendo código Java y SQL.

Jerarquía de Clases



SQL

Estas son las tablas básicas y alguna consulta que he realizado para comprobar los datos de la aplicación.

```
-- Tabla Jugador
CREATE TABLE Jugador (
    jugador_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_inscripcion DATE
);

ALTER TABLE Jugador ADD COLUMN puesto INT DEFAULT 0;
SELECT jugador_id, nombre, puesto FROM Jugador;

describe Jugador;

-- Tabla Torneo
CREATE TABLE Torneo (
    torneo_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE
);

-- Tabla Ronda
CREATE TABLE Ronda (
    ronda_id INT AUTO_INCREMENT PRIMARY KEY,
    torneo_id INT NOT NULL,
    nombre_ronda VARCHAR(50),
    fecha DATE,
    FOREIGN KEY (torneo_id) REFERENCES Torneo(torneo_id)
);

-- Tabla Partida
```

```
CREATE TABLE Partida (
    partida_id INT AUTO_INCREMENT PRIMARY KEY,
    ronda_id INT NOT NULL,
    jugador_blancas_id INT NOT NULL,
    jugador_negras_id INT NOT NULL,
    ganador_id INT,
    estado ENUM('EN_CURSO', 'FINALIZADA', 'CANCELADA') DEFAULT 'EN_CURSO',
    fecha_hora_inicio DATETIME,
    fecha_hora_fin DATETIME,
    FOREIGN KEY (ronda_id) REFERENCES Ronda(ronda_id),
    FOREIGN KEY (jugador_blancas_id) REFERENCES Jugador(jugador_id),
    FOREIGN KEY (jugador_negras_id) REFERENCES Jugador(jugador_id),
    FOREIGN KEY (ganador_id) REFERENCES Jugador(jugador_id)
);

SELECT ronda_id, nombre_ronda FROM Ronda;

SELECT r.ronda_id, r.nombre_ronda, p.partida_id
FROM Ronda r
LEFT JOIN Partida p ON r.ronda_id = p.ronda_id;

SELECT * FROM Ronda WHERE ronda_id = 1;

DESCRIBE Ronda;
```

Java

Paquete CONTROLADOR

Este método me permite simular el torneo, actualizando las tablas y su información. Desactivo ***conn.setAutoCommit*** para que las transacciones de datos no se confirmen automáticamente, dado que si hay algún fallo pueda revertir los cambios con facilidad.

Después creo partidas, rondas y demás utilidades para poder simular el torneo y mostrarlo en JTable.

```
public void simularTorneo() throws SQLException {
    try (Connection conn = GestorBD.conectarBDD()) {
        conn.setAutoCommit(false); // Desactivar auto-commit

        GestorBD.ejecutarUpdate("DELETE FROM Partida");
        GestorBD.ejecutarUpdate("DELETE FROM Ronda");
        GestorBD.ejecutarUpdate("DELETE FROM Jugador");
        GestorBD.ejecutarUpdate("ALTER TABLE Jugador AUTO_INCREMENT = 1");

        List<Jugador> jugadores = insertarJugadoresDePrueba();

        List<Ronda> rondas = new ArrayList<>();
        int numRonda = 1;
        String[] nombresRondas = {"Treintaidósavos", "Dieciseisavos", "Octavos", "Cuartos", "Ser
        Random random = new Random();

        while (jugadores.size() > 1) {
            String nombreRonda = nombresRondas[numRonda - 1];
            Ronda ronda = new Ronda(0, 1, nombreRonda, new Date(System.currentTimeMillis()));
            RondaDAO.crearRonda(ronda);
            System.out.println("Ronda creada - ID: " + ronda.getRondaId() + ", Nombre: " + nombr
            rondas.add(ronda);
```

```
            List<Jugador> ganadores = new ArrayList<>();
            int numPartidas = jugadores.size() / 2;

            for (int i = 0; i < numPartidas; i++) {
                Jugador jugador1 = jugadores.get(i * 2);
                Jugador jugador2 = jugadores.get(i * 2 + 1);

                Jugador ganador = simularPartida(jugador1, jugador2);

                int minutos = 20 + random.nextInt(36);
                String tiempo = String.format("%02d:00", minutos);

                Partida partida = new Partida(
                    jugador1,
                    jugador2,
                    ganador,
                    tiempo,
                    Timestamp.valueOf(LocalDateTime.now()),
                    Timestamp.valueOf(LocalDateTime.now().plusMinutes(minutos)),
                    ronda
                );
                PartidaDAO.crearPartida(partida);

                ganadores.add(ganador);
            }

            jugadores = ganadores;
            numRonda++;
        }



        conn.commit();
        System.out.println("Transacción confirmada.");

        List<Partida> partidasActualizadas = PartidaDAO.obtenerTodasLasPartidas();
        torneoFrame.actualizarRondasEnComboBox();
        torneoFrame.actualizarPartidas(partidasActualizadas);

    } catch (SQLException e) {
        System.err.println("Error en la transacción: " + e.getMessage());
        throw e;
    }
}
```

Observaciones

Para la realización de este ejercicio se ha utilizado JTable dado la comodidad que propociona para tratar datos.

Se ha preferido en un primer momento insertar los datos de prueba en el propio SQL para mayor comodidad.