



DAMA SPORTS

Sports Club



Equipo y roles

Modelo

Métodos principales y reglas

Reglas de negocio

Persistencia

Flujo de Git

Demo

C
R
O

L

E

S

Jose Juan SR

Líder técnico: coordina arquitectura y revisiones

Emiliano José SP

Integración y pruebas: conecta con la UI del profesor, diseña casos de prueba

Aarón PS

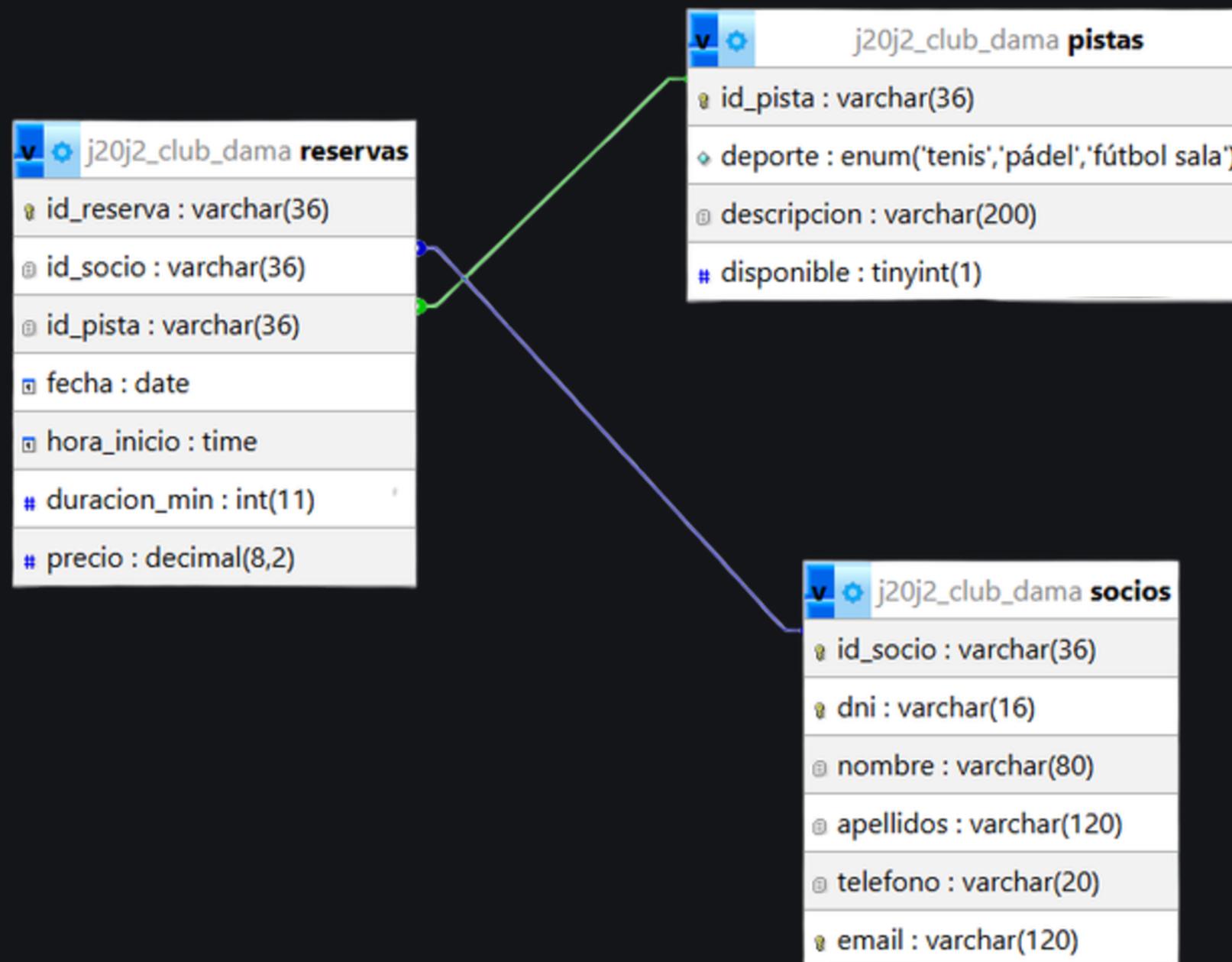
Responsable de Persistencia (JDBC/DAO)

Antonio Jesús PS

Responsable de dominio: reglas de negocio (solapes, disponibilidad, bajas)



Modelo: Diagrama simple





Métodos

Conexión a la
Base de Datos
remota

```
public ClubDeportivo() throws SQLException { 1 usage  & Jose20Juan
    conexion = DriverManager.getConnection(url: "jdbc:mysql://mysql-j20j2.alwaysdata.net:3306/j20j2_club_dama?serverTimezone=UTC",
                                         user: "j20j2", password: "CholoDiversion");
}
```

Método getSocios

```
public ArrayList<Socio> getSocios() throws SQLException { 3 usages  & Jose20Juan
    ArrayList<Socio> listaSocios = new ArrayList<>();

    String sql = "SELECT id_socio, dni, nombre, apellidos, telefono, email\n" +
                 "FROM socios;\n";
    PreparedStatement pst = conexion.prepareStatement(sql);
    ResultSet rs = pst.executeQuery();

    while (rs.next()) {
        Socio socio = new Socio(rs.getString(columnIndex: 1), rs.getString(columnIndex: 2),
                               rs.getString(columnIndex: 3), rs.getString(columnIndex: 4),
                               rs.getString(columnIndex: 5), rs.getString(columnIndex: 6));
        listaSocios.add(socio);
    }

    rs.close();

    return listaSocios;
}
```

Método getPistas

```
public ArrayList<Pista> getPistas() throws SQLException { 3 usages  & Jose20Juan
    ArrayList<Pista> listaPistas = new ArrayList<>();

    String sql = "SELECT id_pista, deporte, descripcion, disponible\n" +
                 "FROM pistas;\n";
    PreparedStatement pst = conexion.prepareStatement(sql);
    ResultSet rs = pst.executeQuery();

    while (rs.next()) {
        Pista pista = new Pista(rs.getString(columnIndex: 1), rs.getString(columnIndex: 2),
                               rs.getString(columnIndex: 3), rs.getBoolean(columnIndex: 4));
        listaPistas.add(pista);
    }

    rs.close();

    return listaPistas;
}
```

Método getReservas

```
public ArrayList<Reserva> getReservas() throws SQLException { 2 usages  & Jose20Juan
    ArrayList<Reserva> listaReservas = new ArrayList<>();

    String sql = "SELECT id_reserva, id_socio, id_pista, fecha, hora_inicio, duracion_min, precio\n" +
                 "FROM reservas;\n";
    PreparedStatement pst = conexion.prepareStatement(sql);
    ResultSet rs = pst.executeQuery();

    while (rs.next()) {
        Reserva reserva = new Reserva(rs.getString(columnIndex: 1), rs.getString(columnIndex: 2),
                                     rs.getString(columnIndex: 3), rs.getDate(columnIndex: 4).toLocalDate(), rs.getTime(columnIndex: 5).toLocalTime(),
                                     rs.getInt(columnIndex: 6), rs.getDouble(columnIndex: 7));
        listaReservas.add(reserva);
    }

    rs.close();

    return listaReservas;
}
```



Métodos

Método altaSocio

```
public boolean altaSocio(Socio socio) throws SQLException { 1 usage  
  
    String sql = "INSERT INTO socios\n" +  
        "(id_socio, dni, nombre, apellidos, telefono, email)\n" +  
        "VALUES (?, ?, ?, ?, ?, ?);  
  
    PreparedStatement pst = conexion.prepareStatement(sql);  
  
    pst.setString(parameterIndex: 1, socio.getIdSocio());  
    pst.setString(parameterIndex: 2, socio.getDni());  
    pst.setString(parameterIndex: 3, socio.getNombre());  
    pst.setString(parameterIndex: 4, socio.getApellidos());  
    pst.setString(parameterIndex: 5, socio.getTelefono());  
    pst.setString(parameterIndex: 6, socio.getEmail());  
  
    pst.executeUpdate();  
    return true;  
}
```

Método altaPista

```
public boolean altaPista(Pista pista) throws SQLException { 1 usage  
  
    String sql = "INSERT INTO pistas\n" +  
        "(id_pista, deporte, descripcion, disponible)\n" +  
        "VALUES (?, ?, ?, ?);  
  
    PreparedStatement pst = conexion.prepareStatement(sql);  
  
    pst.setString(parameterIndex: 1, pista.getIdPista());  
    pst.setString(parameterIndex: 2, pista.getDeporte());  
    pst.setString(parameterIndex: 3, pista.getDescripcion());  
    pst.setBoolean(parameterIndex: 4, pista.isDisponible());  
  
    pst.executeUpdate();  
    return true;  
}
```

Método crearReserva

```
public boolean crearReserva(Reserva reserva) throws SQLException { 1 usage  
  
    String sql = "{CALL sp_crear_reserva(?, ?, ?, ?, ?, ?)}";  
  
    CallableStatement cst = conexion.prepareCall(sql);  
  
    cst.setString(parameterIndex: 1, reserva.getIdReserva());  
    cst.setString(parameterIndex: 2, reserva.getIdSocio());  
    cst.setString(parameterIndex: 3, reserva.getIdPista());  
    cst.setDate(parameterIndex: 4, Date.valueOf(reserva.getFecha()));  
    cst.setTime(parameterIndex: 5, Time.valueOf(reserva.getHoraInicio()));  
    cst.setInt(parameterIndex: 6, reserva.getDuracionMin());  
  
    cst.execute();  
    return true;  
}
```



Métodos

Método bajaPista

```
public boolean bajaPista(Socio socio) throws SQLException {  
  
    String sql = "DELETE FROM socios WHERE id_socio = ?";  
  
    PreparedStatement pst = conexion.prepareStatement(sql);  
  
    pst.setString(parameterIndex: 1, socio.getIdSocio());  
  
    pst.executeUpdate();  
  
    return true;  
}
```

Método cambiarDisponibilidad

```
public void cambiarDisponibilidad(String nombrePista, boolean disponible) throws SQLException {  
    String sql = "UPDATE pistas SET disponible = ? WHERE id_pista = ?";  
  
    PreparedStatement pst = conexion.prepareStatement(sql);  
  
    pst.setBoolean(parameterIndex: 1, disponible);  
  
    pst.setString(parameterIndex: 2, nombrePista);  
  
    int filasAfectadas = pst.executeUpdate();  
  
    pst.close();  
  
    System.out.println("Se actualizaron " + filasAfectadas + " filas.");  
}
```

Método crearReserva

```
public boolean cancelarReserva(Reserva reserva) throws SQLException {  
  
    String sql = "DELETE FROM reservas WHERE id_reserva = ?";  
  
    PreparedStatement pst = conexion.prepareStatement(sql);  
    pst.setString(parameterIndex: 1, reserva.getIdReserva());  
    pst.executeUpdate();  
  
    return true;  
}
```



Reglas de negocio

Disponibilidad: Antes de crear una reserva, el sistema verifica que este disponible.

No solapes: Un socio no podrá reservas mas de una pista al mismo tiempo.

Baja socio con reservas futuras: Si un socio quiere darse de baja pero tiene una reserva el sistema lo evitará.

Tiempos validos: Las reservas deben tener una fecha y una hora actual.



Persistencia

Persistencia a bases de datos

Table	Action	Rows	Type	Collation	Size	Overhead
pistas	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
reservas	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	48.0 KiB	-
socios	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	48.0 KiB	-
3 tables Sum			9 InnoDB utf8mb4_general_ci 112.0 KiB			0 B

Tabla reservas

	Edit	Copy	Delete	R-001	S-001	P-002	2025-11-23	10:00:00	60	12.00
	Edit	Copy	Delete	R-002	S-003	P-001	2025-11-23	10:00:00	60	12.00

Tabla socios

	Edit	Copy	Delete	S-001	12345678A	Ana	López	600111222	ana@example.com
	Edit	Copy	Delete	S-002	87654321B	Luis	García	600333444	luis@example.com
	Edit	Copy	Delete	S-003	123456789	Martín	Martínez Martínez	678354123	martin@martin.com

Tabla pistas

	Edit	Copy	Delete	P-001	tenis	Tenis exterior	1
	Edit	Copy	Delete	P-002	pádel	Pádel cubierto	1
	Edit	Copy	Delete	P-003	fútbol sala	Pabellón	0
	Edit	Copy	Delete	P-004	tenis	Tierra batida	0



Flujo de Git

```
PS C:\Users\josej\Desktop\dam-ads-u2-equipo007> git log --graph --oneline --all --decorate
* 25d4f01 (HEAD -> main, origin/main, origin/HEAD) -
* aa44af8 -
* 838f2ff Algún ajuste más
* ebc3f5c Más pequeños ajustes hechos
* 20ca00a Merge pull request #4 from Jose20Juan/Antonio-Jesus
|\
| * b6851f6 (origin/Antonio-Jesus) Merge branch 'main' into Antonio-Jesus
| |
| |
| *
| * 2de64c9 Arreglo de los botones
| * a123f6c metodo baja socio y botones
| /
| * ada9b3e Metodo cancelarReserva implementado
| \
| * eeed1a7 (origin/Emi) Metodo cancelarReserva implementado
| /
| * 4500a39 Metodo CambiarDisponibilidad implementado
| \
| * b4a6b3a (origin/Aaron) Metodo CambiarDisponibilidad implementado
| /
| * d58eb4c Merge pull request #1 from Jose20Juan/JJ
| \
| * eff54ae (origin/JJ, JJ) Añadido altaSocio, altaPista, crearReserva
| * 7b238d3 Añado gets de socios, pistas y reservas
| /
| * bba2db7 .
| * 0dca226 -
| * 01e6440 Conexión a la base de datos remota
| * 2cc3914 -
| * bbca80e Ahora sí, primer commit
| * ff33a1f Estaba mal lo otro
| * d8ae44c Commit inicial
| * 4823129 Commit inicial
:... skipping...
```



Demo

Club DAMA Sports

Archivo Socios Pistas Reservas Ver

idSocio*	4
DNI	48750911X
Nombre	Demo
Apellidos	Demo
Teléfono	655 555 555
Email	demo@gmail.com

Mensaje

Socio insertado correctamente

Listo

Club DAMA Sports

Archivo Socios Pistas Reservas Ver

Resumen
Socios

ID	Nombre
4	Demo
S-001	Ana
S-002	Luis
S-003	Martín

Pistas

ID	Deporte	Disponible
P-001	tenis	true
P-002	pádel	true
P-003	fútbol sala	false
P-004	tenis	false

ID	Socio	Pista	Fecha	Inicio	Min	Precio
R-001	S-001	P-002	2025-11-23	10:00	60	12.0 €
R-002	S-003	P-001	2025-11-23	10:00	60	12.0 €

Demostración de la alta de un socio y su creación.



Demo

The screenshot shows a MySQL Workbench interface with a table named 'socio'. The table has columns: id_socio, dni, nombre, apellidos, telefono, and email. The data is as follows:

	id_socio	dni	nombre	apellidos	telefono	email
<input type="checkbox"/>	Edit Copy Delete	4	48750911X	Demo	655 555 555	demo@gmail.com
<input type="checkbox"/>	Edit Copy Delete	S-001	12345678A	Ana López	600111222	ana@example.com
<input type="checkbox"/>	Edit Copy Delete	S-002	87654321B	Luis García	600333444	luis@example.com
<input type="checkbox"/>	Edit Copy Delete	S-003	123456789	Martín Martínez	678354123	martin@martin.com

Below the table, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'. At the bottom, there are buttons for 'Print', 'Copy to clipboard', 'Export', 'Display chart', 'Create view', and a 'Bookmark this SQL query' button.

Carga completa en la base de datos



Demo

The screenshot shows a Windows application window titled "Club DAMA Sports". The menu bar includes "Archivo", "Socios", "Pistas", "Reservas", and "Ver". The "Socios" tab is selected, showing a table of members:

ID	Nombre
4	Demo
S-001	Ana
S-002	Luis
S-003	Martín

To the right of this table is another table titled "Pistas" (Courts):

ID	Deporte	Disponible
P-001	tenis	true
P-002	pádel	true
P-003	fútbol sala	false
P-004	tenis	false

Below the member table, a message box titled "Error" displays the message "Socio no Borrado correctamente" (Member not deleted correctly). A button labeled "Aceptar" (Accept) is visible at the bottom of the dialog.

Demostración de como no se puede dar de baja un socio si tiene una reserva activa.

C

FINAL