

Base De Datos Final



Germán Martínez Calvente, 2025

Atlantida Formación profesional

Desarrollo de Aplicaciones Multiplataforma

Bases de Datos

Fernando Aranda García

Introducción y Objetivo del Proyecto

El presente trabajo se enfoca en el diseño e implementación de una base de datos relacional para una aplicación llamada "Pokémon TCG Pocket", orientada a la gestión de cartas, mazos y partidas del juego de cartas colecciónables Pokémon (TCG).

El objetivo principal es crear una base de datos estructurada, normalizada y relacional que permita:

- Registrar usuarios (jugadores) del sistema.
- Almacenar cartas con sus propiedades y estadísticas de juego.
- Gestionar mazos personalizados creados por los jugadores.
- Llevar un historial de partidas jugadas y sus resultados.

Justificación del Diseño y Utilidad de la Base de Datos

La base de datos propuesta permite centralizar y organizar eficientemente toda la información relevante para la gestión de un sistema de TCG, asegurando integridad, coherencia y facilidad de acceso a los datos.

Su utilidad se ve reflejada en diversos escenarios:

- Los jugadores pueden crear mazos personalizados y registrar partidas contra otros usuarios.
- El sistema puede mostrar estadísticas de cartas o partidas jugadas para análisis.
- Facilita futuras integraciones con una interfaz web, una app móvil o un sistema de torneos en línea.

El modelo también está diseñado para ser escalable y modificable, permitiendo agregar nuevas funciones (como logros, comentarios, rankings, etc.) sin alterar las estructuras centrales

Análisis y Descripción de las Entidades y Atributos

Jugadores

Representa a los usuarios registrados en el sistema.

- **id_jugador:** Identificador único del jugador.
 - **nombre_usuario:** Alias o nick del jugador.
 - **email:** Correo electrónico (único).
 - **fecha_registro:** Fecha de creación del perfil.
-

Cartas

Almacena información básica de cada carta del juego.

- **id_carta:** Identificador único.
 - **nombre, tipo, sub_tipo, rareza, expansion_poke:** Atributos que describen la carta dentro del universo del TCG.
-

Estadísticas de cartas

Contiene datos técnicos del rendimiento de una carta durante el juego.

- **id_carta:** Referencia a la carta.
- **hp, ataques, daños, debilidad, resistencia, coste_retiro:** Elementos necesarios para el uso real en partida.

Mazos

Permite a los jugadores crear sus propios mazos.

- **num_mazo:** Identificador del mazo.
 - **nombre_mazo:** Nombre personalizado del mazo.
 - **num_jugador:** Relación con el jugador que lo creó.
 - **fecha_creacion:** Fecha en la que se creó el mazo.
-

Cartas en mazo

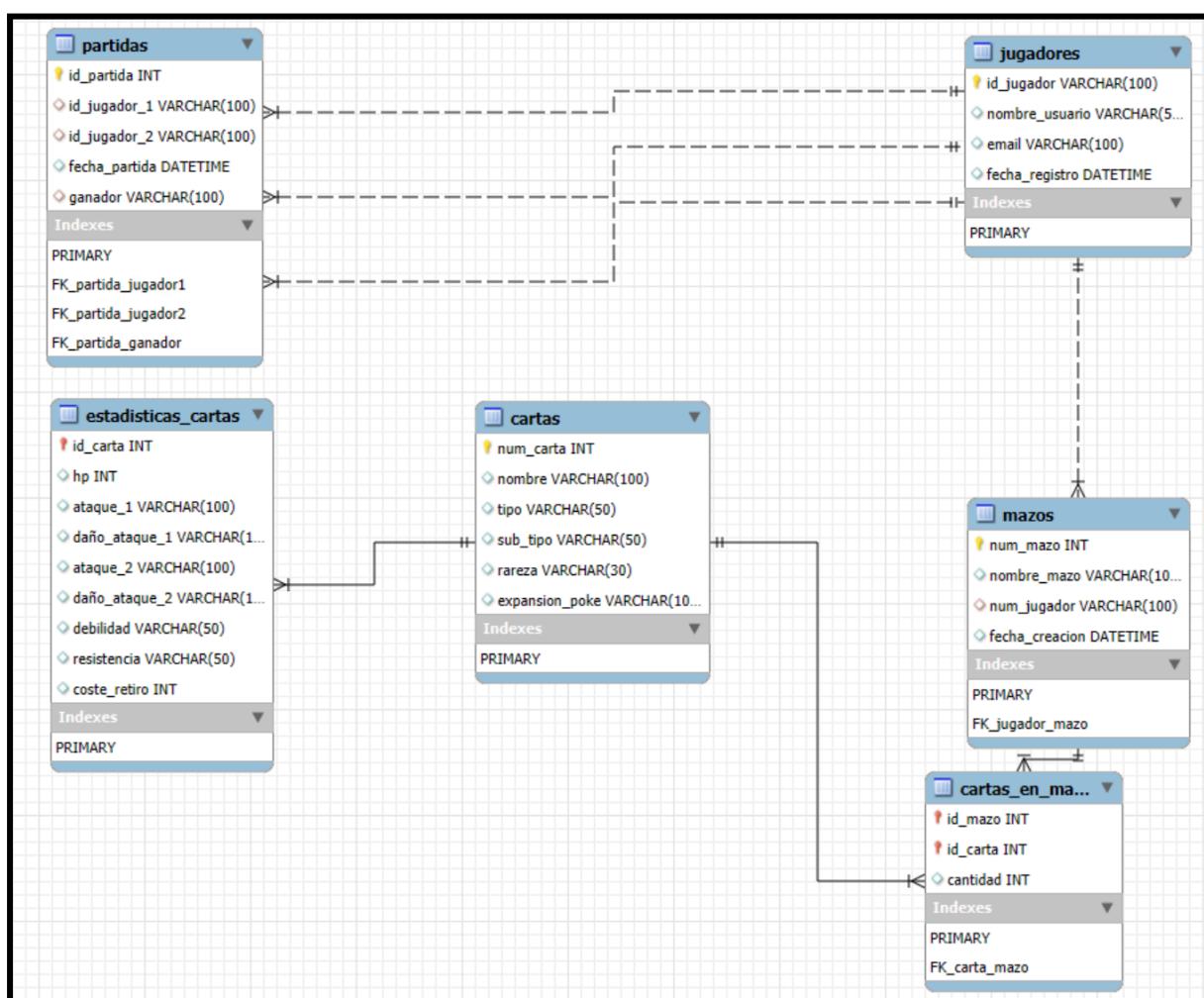
Es una tabla intermedia que modela la relación N:M entre cartas y mazos.

- **id_mazo:** Mazo donde está incluida la carta.
 - **id_carta:** Carta incluida en el mazo.
 - **cantidad:** Número de veces que esa carta aparece en el mazo.
-

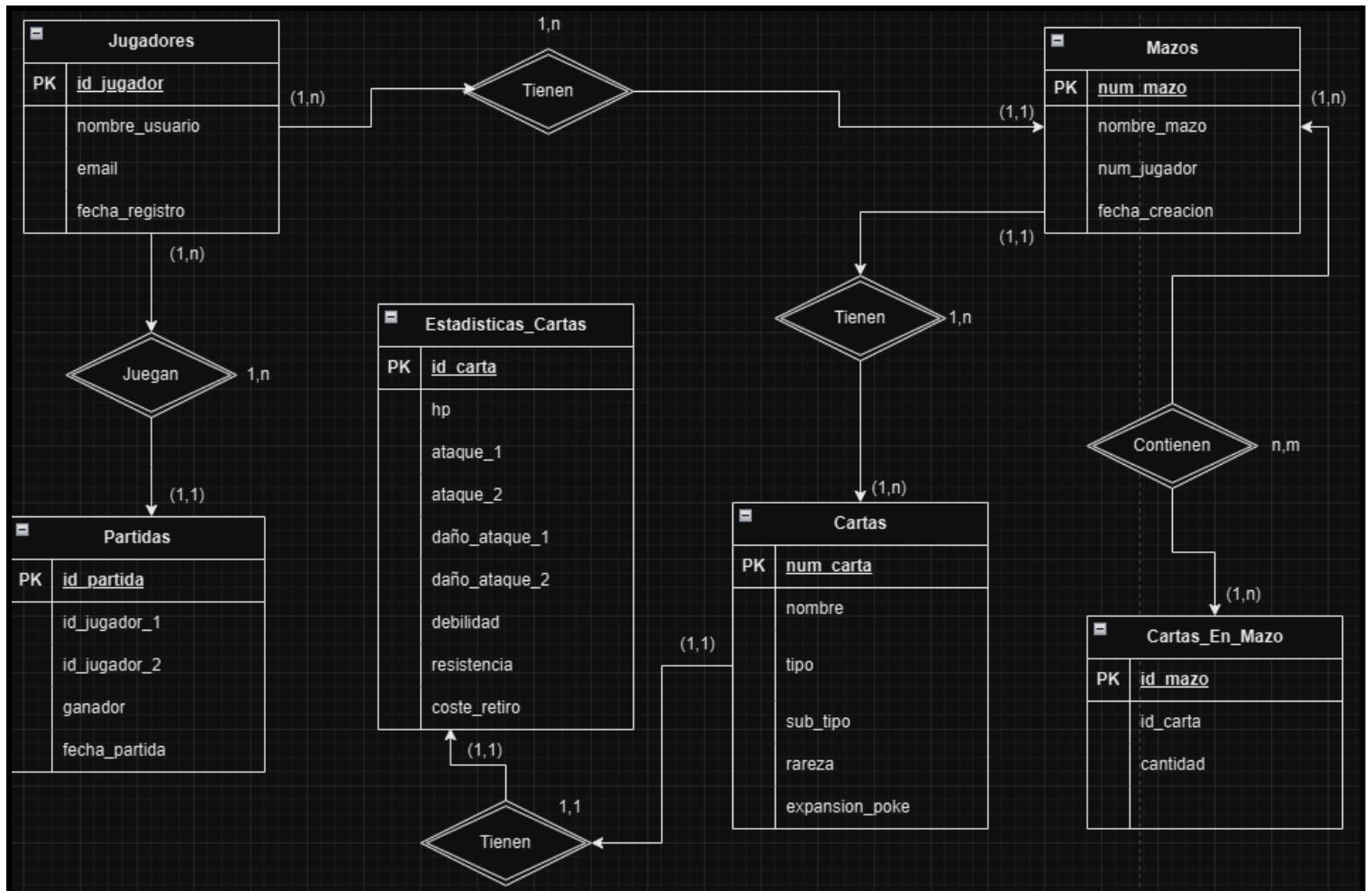
Partidas

Registra las partidas entre jugadores.

- **id_partida:** Identificador único de la partida.
- **id_jugador_1, id_jugador_2:** Jugadores que participaron.
- **ganador:** Jugador que ganó.
- **fecha_partida:** Fecha en que se jugó.



DER Y NORMALIZACIÓN



JUGADORES (id_jugador, num_mazo, id_partida, nombre_usuario, email, fecha_registro)

CARTAS(num_carta, num_mazo, nombre, tipo, sub_tipo, rareza, expansion_poke)

ESTADISTICAS_CARTAS (id_carta, hp, ataque_1, daño_ataque_1, ataque_2, daño_ataque_2, debilidad, resistencia, coste_retiro)

MAZOS (num_mazo, nombre_mazo, num_jugador, fecha_creacion)

CARTAS_EN_MAZO (id_mazo, id_carta, cantidad)

PARTIDAS (id_partida, id_jugador_1, id_jugador_2, fecha_partida, ganador)

MAZOS_CARTAS_EN_MAZO (num_mazo, id_mazo)

TABLAS BD

```
CREATE DATABASE pokemon_tcg_pocket;
```

```
USE pokemon_tcg_pocket;
```

```
CREATE TABLE jugadores (
    id_jugador VARCHAR(100) PRIMARY KEY,
    nombre_usuario VARCHAR(50),
    email VARCHAR(100),
    fecha_registro DATETIME
);
```

```
CREATE TABLE cartas (
    num_carta INT PRIMARY KEY,
    nombre VARCHAR(100),
    tipo VARCHAR(50),
    sub_tipo VARCHAR(50),
    rareza VARCHAR(30),
    expansion_poke VARCHAR(100)
);
```

```
CREATE TABLE estadisticas_cartas (
    id_carta INT PRIMARY KEY,
    hp INT,
    ataque_1 VARCHAR(100),
    daño_ataque_1 VARCHAR(10),
    ataque_2 VARCHAR(100),
    daño_ataque_2 VARCHAR(10),
    debilidad VARCHAR(50),
    resistencia VARCHAR(50),
    coste_retiro INT
);
```

```
CREATE TABLE mazos (
    num_mazo INT PRIMARY KEY,
    nombre_mazo VARCHAR(100),
    num_jugador VARCHAR(100),
    fecha_creacion DATETIME
);
```

```
CREATE TABLE cartas_en_mazo (
    id_mazo INT,
```

```

id_carta INT,
cantidad INT,
PRIMARY KEY (id_mazo, id_carta));
CREATE TABLE partidas (
id_partida INT PRIMARY KEY,
id_jugador_1 VARCHAR(100),
id_jugador_2 VARCHAR(100),
fecha_partida DATETIME,
ganador VARCHAR(100)
);
ALTER TABLE estadisticas_cartas
ADD CONSTRAINT FK_carta_estadisticas
FOREIGN KEY (id_carta) REFERENCES cartas(num_carta)
ON DELETE CASCADE ON UPDATE CASCADE;

-- Relaciona mazos con jugadores
ALTER TABLE mazos
ADD CONSTRAINT FK_jugador_mazo
FOREIGN KEY (num_jugador) REFERENCES jugadores(id_jugador)
ON DELETE SET NULL ON UPDATE CASCADE;

-- Relaciona cartas_en_mazo con mazos
ALTER TABLE cartas_en_mazo
ADD CONSTRAINT FK_mazo_cartas
FOREIGN KEY (id_mazo) REFERENCES mazos(num_mazo)
ON DELETE CASCADE ON UPDATE CASCADE;

-- Relaciona cartas_en_mazo con cartas
ALTER TABLE cartas_en_mazo
ADD CONSTRAINT FK_carta_mazo
FOREIGN KEY (id_carta) REFERENCES cartas(num_carta)
ON DELETE CASCADE ON UPDATE CASCADE;

-- Relaciona partidas con jugadores (jugador 1)
ALTER TABLE partidas
ADD CONSTRAINT FK_partida_jugador1
FOREIGN KEY (id_jugador_1) REFERENCES jugadores(id_jugador)
ON DELETE SET NULL ON UPDATE CASCADE;

-- Relaciona partidas con jugadores (jugador 2)
ALTER TABLE partidas
ADD CONSTRAINT FK_partida_jugador2
FOREIGN KEY (id_jugador_2) REFERENCES jugadores(id_jugador)
ON DELETE SET NULL ON UPDATE CASCADE;

-- Relaciona partidas con el ganador (también jugador)
ALTER TABLE partidas
ADD CONSTRAINT FK_partida_ganador

```

```
FOREIGN KEY (ganador) REFERENCES jugadores(id_jugador)
ON DELETE SET NULL ON UPDATE CASCADE;
```

```
INSERT INTO jugadores (id_jugador, nombre_usuario, email, fecha_registro) VALUES
('JUG001', 'AshKetchum', 'ash@pallet.com', '2023-03-01 10:00:00'),
('JUG002', 'MistyWater', 'misty@cerulean.com', '2023-03-05 15:30:00'),
('JUG003', 'BrockSolid', 'brock@pewter.com', '2023-03-10 08:45:00'),
('JUG004', 'GaryOak', 'gary@viridian.com', '2023-03-12 13:20:00');
```

```
INSERT INTO cartas (id_carta, nombre, tipo, sub_tipo, rareza, expansion_poke) VALUES
(1, 'Pikachu', 'Eléctrico', 'Básico', 'Común', 'Espada y Escudo'),
(2, 'Charizard', 'Fuego', 'Etapa 2', 'Ultra Rara', 'Destino Brillante'),
(3, 'Bulbasaur', 'Planta', 'Básico', 'Común', 'Orígenes'),
(4, 'Squirtle', 'Agua', 'Básico', 'Común', 'Orígenes'),
(5, 'Meowth', 'Normal', 'Básico', 'Rara', 'Rebel Clash'),
(6, 'Gengar', 'Fantasma', 'Etapa 2', 'Rara Holo', 'Oscuridad Incandescente'),
(7, 'Lucario', 'Lucha', 'Etapa 1', 'Rara', 'Evoluciones'),
(8, 'Eevee', 'Normal', 'Básico', 'Común', 'Truenos Perdidos');
```

```
INSERT INTO estadisticas_cartas (id_carta, hp, ataque_1, daño_ataque_1, ataque_2,
daño_ataque_2, debilidad, resistencia, coste_retiro) VALUES
(1, 60, 'Impactrueno', 20, 'Chispa', 30, 'Tierra', 'Acero', 1),
(2, 150, 'Lanzallamas', 90, 'Explosión Fuego', 130, 'Agua', 'Planta', 3),
(3, 50, 'Látigo Cepa', 20, 'Drenadoras', 20, 'Fuego', 'Agua', 1),
(4, 50, 'Pistola Agua', 20, 'Burbuja', 10, 'Eléctrico', 'Fuego', 1),
(5, 70, 'Placaje', 10, 'Arañazo', 20, 'Lucha', 'Fantasma', 4),
(6, 120, 'Lengua Maldita', 60, 'Sombras Tenebrosas', 90, 'Psíquico', 'Lucha', 2),
(7, 100, 'Puño Certeño', 40, 'Aguijón', 40, 'Psíquico', 'Oscuro', 2),
(8, 60, 'Golpe Rápido', 30, 'Mordisco', 20, 'Lucha', 'Ninguna', 1);
```

```
INSERT INTO mazos (num_mazo, nombre_mazo, num_jugador, fecha_creacion) VALUES
(101, 'Rayo Explosivo', 'JUG001', '2023-03-06 12:00:00'),
(102, 'Fuego Ardiente', 'JUG004', '2023-03-12 14:00:00'),
(103, 'Verde Natural', 'JUG002', '2023-03-08 09:00:00'),
(104, 'Agua Letal', 'JUG002', '2023-03-09 10:15:00');
```

```
INSERT INTO cartas_en_mazo (id_mazo, id_carta, cantidad) VALUES
(101, 1, 3), -- Pikachu
(101, 5, 2), -- Snorlax
(101, 8, 2), -- Eevee

(102, 2, 3), -- Charizard
(102, 5, 1), -- Snorlax
(102, 7, 2), -- Lucario

(103, 3, 2), -- Bulbasaur
(103, 8, 2), -- Eevee
```

```
(104, 4, 4), -- Squirtle
(104, 6, 2), -- Gengar
(104, 1, 1); -- Pikachu
```

```
INSERT INTO partidas (id_partida, id_jugador_1, id_jugador_2, fecha_partida, ganador)
VALUES
(201, 'JUG001', 'JUG004', '2023-03-15 17:00:00', 'JUG004'),
(202, 'JUG002', 'JUG003', '2023-03-16 18:00:00', 'JUG002'),
(203, 'JUG003', 'JUG002', '2023-03-17 16:00:00', 'JUG003'),
(204, 'JUG003', 'JUG004', '2023-03-18 19:00:00', 'JUG003');
```

ACTUALIZACIONES

```
UPDATE jugadores
SET nombre_usuario = 'AshLegend'
WHERE id_jugador = 'JUG001';
```

```
UPDATE estadisticas_cartas
SET hp = 160
WHERE id_carta = 2;
```

```
UPDATE mazos
SET nombre_mazo = 'Fuego Imparable'
WHERE num_mazo = 102;
```

BORRADOS

```
DELETE FROM cartas_en_mazo
WHERE id_mazo = 103 AND id_carta = 8;
```

```
DELETE FROM mazos
WHERE num_mazo = 104;
```

```
DELETE FROM jugadores
WHERE id_jugador = 'JUG003';
```

CONSULTAS BÁSICAS

```
SELECT nombre_usuario, email, fecha_registro
```

```

FROM jugadores;

SELECT cartas.nombre, estadisticas_cartas.hp
FROM cartas
JOIN estadisticas_cartas ON cartas.id_carta = estadisticas_cartas.id_carta
WHERE estadisticas_cartas.hp > 100;

SELECT mazos.nombre_mazo, mazos.fecha_creacion
FROM mazos
JOIN jugadores ON mazos.num_jugador = jugadores.id_jugador
WHERE jugadores.nombre_usuario = 'AshLegend';

```

PROCEDIMIENTO ALMACENADO

```

1   DELIMITER //
2
3   CREATE PROCEDURE insertar_partida (
4       IN p_id INT,
5       IN p_jugador1 VARCHAR(100),
6       IN p_jugador2 VARCHAR(100),
7       IN p_fecha DATETIME,
8       IN p_ganador VARCHAR(100)
9   )
10  BEGIN
11      INSERT INTO partidas (id_partida, id_jugador_1, id_jugador_2, fecha_partida, ganador)
12          VALUES (p_id, p_jugador1, p_jugador2, p_fecha, p_ganador);
13  END //
14
15  DELIMITER ;
16
17 • CALL insertar_partida(7, 'jug001', 'jug002', NOW(), 'jug001');
18
19 • SELECT * FROM partidas ORDER BY id_partida DESC;
20

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id_partida	id_jugador_1	id_jugador_2	fecha_partida	ganador
▶	204	JUG003	JUG004	2023-03-18 19:00:00	JUG003
	203	JUG001	JUG003	2023-03-17 16:00:00	JUG002
	202	JUG002	JUG003	2023-03-16 18:30:00	JUG003
	201	JUG001	JUG004	2023-03-15 17:00:00	JUG004
	7	jug001	jug002	2025-05-16 12:09:38	jug001
	1	jug001	jug002	2025-05-16 12:06:33	jug001
*	NULL	NULL	NULL	NULL	NULL

partidas 2 ×

Output :

Action Output

#	Time	Action
✓	1 12:09:37	CREATE PROCEDURE insertar_partida (IN p_id INT, IN p_jugador1 VARCHAR(100), IN p_jugador2 VARCHAR(100))
✓	2 12:09:38	CALL insertar_partida(7, 'jug001', 'jug002', NOW(), 'jug001')
✓	3 12:09:38	SELECT * FROM partidas ORDER BY id_partida DESC LIMIT 0, 50000

El NOW() funciona para poner la fecha actual, MOLA.

CON VARIABLES LOCALES

```

1  DELIMITER
2
3  CREATE PROCEDURE contar_cartas_en_mazo(IN mazo_id INT)
4  BEGIN
5      SET @total_cartas := 0;
6
7      SELECT SUM(cantidad)
8          INTO @total_cartas
9          FROM cartas_en_mazo
10         WHERE id_mazo = mazo_id;
11
12         SELECT CONCAT('Total de cartas en el mazo nº', mazo_id, ': ', @total_cartas) AS resultado;
13     END
14
15     DELIMITER ;
16
17 • CALL contar_cartas_en_mazo(101);

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

resultado
Total de cartas en el mazo nº101: 7

Result 3 x

Output :::::

Action Output

#	Time	Action
1	12:30:59	CREATE PROCEDURE contar_cartas_en_mazo(IN mazo_id INT) BEGIN -- Variable de sesión SET @total_cartas := 0; -- C
2	12:31:18	CALL contar_cartas_en_mazo(101)

FUNCION SQL

```

1   DELIMITER //
2
3 •  CREATE FUNCTION total_mazos_por_jugador(p_id_jugador VARCHAR(100))
4     RETURNS INT
5
6   BEGIN
7     DECLARE total INT;
8
9     SELECT COUNT(*) INTO total
10    FROM mazos
11    WHERE num_jugador = p_id_jugador;
12
13    RETURN total;
14 END //
15
16 DELIMITER ;
17
18 •  SELECT total_mazos_por_jugador('jug001') AS total_mazos;
19

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
total_mazos				1

Result 2 x

Output

Action Output

#	Time	Action
1	12:16:57	CREATE FUNCTION total_mazos_por_jugador(p_id_jugador VARCHAR(100)) RETURNS INT
2	12:16:57	SELECT total_mazos_por_jugador('jug001') AS total_mazos LIMIT 0, 50000

CON IF

```

3   CREATE FUNCTION clasificacion_jugador(jugador_id VARCHAR(100))
4     RETURNS VARCHAR(50)
5   BEGIN
6       DECLARE total_victorias INT;
7       DECLARE clasificacion VARCHAR(50);
8
9       SELECT COUNT(*) INTO total_victorias
10      FROM partidas
11      WHERE ganador = jugador_id;
12
13      IF total_victorias >= 3 THEN
14          SET clasificacion = 'Experto';
15      ELSEIF total_victorias = 2 THEN
16          SET clasificacion = 'Intermedio';
17      ELSE
18          SET clasificacion = 'Novato';
19      END IF;
20
21      RETURN clasificacion;
22  END
23  DELIMITER ;
24 •  SELECT clasificacion_jugador('JUG003') AS nivel;
25

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	nivel
▶	Intermedio

Result 4 ×

Output ::::::::::::::::::::

Action Output

#	Time	Action
1	12:35:05	CREATE FUNCTION clasificacion_jugador(jugador_id VARCHAR(100)) RETURNS V
2	12:35:19	SELECT clasificacion_jugador('JUG003') AS nivel LIMIT 0, 50000

CONSULTAS

- ```
SELECT id_jugador, nombre_usuario, email, fecha_registro
FROM jugadores
ORDER BY fecha_registro ASC;
```

-- Lista todos los jugadores en el orden en que se registraron
  
- ```
SELECT cartas.nombre, cartas.tipo, cartas_en_mazo.cantidad
FROM cartas_en_mazo
JOIN cartas ON cartas_en_mazo.id_carta = cartas.num_carta
JOIN mazos ON cartas_en_mazo.id_mazo = mazos.num_mazo
WHERE mazos.nombre_mazo = 'Fuego Ardiente';
-- Muestra las cartas incluidas en el mazo llamado "Fuego Ardiente"
```

- ```
SELECT cartas.nombre, estadisticas_cartas.hp
FROM cartas
JOIN estadisticas_cartas ON cartas.num_carta = estadisticas_cartas.id_carta
WHERE estadisticas_cartas.hp > 100;
-- Filtra las cartas que tienen más de 100 puntos de salud
```
  
- ```
SELECT mazos.nombre_mazo, SUM(cartas_en_mazo.cantidad) AS total_cartas
FROM cartas_en_mazo
JOIN mazos ON cartas_en_mazo.id_mazo = mazos.num_mazo
GROUP BY mazos.nombre_mazo;
```

-- Cuenta cuántas cartas tiene cada mazo en total

- ```
SELECT ganador AS id_jugador, COUNT(*) AS victorias
FROM partidas
GROUP BY ganador
ORDER BY victorias DESC;
```

-- Muestra cuántas veces ha ganado cada jugador, de mayor a menor

## SUBCONSULTAS

```
1 • SELECT nombre_usuario
2 FROM jugadores
3 WHERE id_jugador = (
4 SELECT ganador
5 FROM partidas
6 GROUP BY ganador
7 ORDER BY COUNT(*) DESC
8 LIMIT 1
9);
10 -- Devuelve el nombre del jugador con más victorias
11
12 • SELECT nombre_mazo
13 FROM mazos
14 WHERE num_jugador = (
15 SELECT id_jugador
16 FROM jugadores
17 WHERE email = 'misty@cerulean.com'
18);
19 -- Lista los mazos de Misty por su correo
20
21 • SELECT nombre
22 FROM cartas
23 WHERE num_carta = (
24 SELECT id_carta
25 FROM estadisticas_cartas
26 ORDER BY CAST(daño_ataque_1 AS UNSIGNED) DESC
27 LIMIT 1
28);
29 -- Devuelve el nombre de la carta con más daño en su primer ataque
```

```

SELECT nombre_usuario
FROM jugadores
WHERE id_jugador IN (
 SELECT DISTINCT ganador
 FROM partidas
);
-- Lista los nombres de los jugadores que han ganado alguna vez

SELECT cartas.nombre, cartas.tipo
FROM cartas
JOIN cartas_en_mazo ON cartas.num_carta = cartas_en_mazo.id_carta
WHERE cartas_en_mazo.id_mazo = (
 SELECT id_mazo
 FROM cartas_en_mazo
 GROUP BY id_mazo
 ORDER BY SUM(cantidad) DESC
 LIMIT 1
);
-- Muestra las cartas del mazo más grande

```

## TRIGGERS

```

DELIMITER //

• CREATE TRIGGER fecha_creacion_mazo
BEFORE INSERT ON mazos
FOR EACH ROW
BEGIN
 IF NEW.fecha_creacion IS NULL THEN
 SET NEW.fecha_creacion = NOW();
 END IF;
END //

DELIMITER ;

• -- Si alguien intenta crear un mazo y deja el campo fecha_creacion vacío,
-- este trigger lo rellena automáticamente con la fecha actual.

```

```

DELIMITER //

CREATE TRIGGER asignar_ganador_por_defecto
BEFORE INSERT ON partidas
FOR EACH ROW
BEGIN
 IF NEW.id_jugador_2 IS NULL THEN
 SET NEW.ganador = NEW.id_jugador_1;
 END IF;
END //

DELIMITER ;

```

-- Si el jugador 2 no existe (abandonó la partida),  
-- se asigna automáticamente como ganador al jugador 1.

```

DELIMITER //

CREATE TRIGGER rareza_por_defecto
BEFORE INSERT ON cartas
FOR EACH ROW
BEGIN
 IF NEW.rareza IS NULL OR NEW.rareza = '' THEN
 SET NEW.rareza = 'Desconocida';
 END IF;
END //

```

DELIMITER ;

\* -- Cuando la rareza no se ha especificado se añade por defecto rareza "Desconocida"

## Sistema de Partición Básico

```

1 • CREATE TABLE partidas (
2 id_partida INT NOT NULL,
3 id_jugador_1 VARCHAR(100),
4 id_jugador_2 VARCHAR(100),
5 fecha_partida DATETIME NOT NULL,
6 ganador VARCHAR(100),
7 PRIMARY KEY (id_partida, fecha_partida)
8)
9 • PARTITION BY RANGE (YEAR(fecha_partida)) (
10 PARTITION p2023 VALUES LESS THAN (2024),
11 PARTITION p2024 VALUES LESS THAN (2025),
12 PARTITION p2025 VALUES LESS THAN (2026),
13 PARTITION pmax VALUES LESS THAN MAXVALUE
14);
15
16 • INSERT INTO partidas (id_partida, id_jugador_1, id_jugador_2, fecha_partida, ganador) VALUES
17 (1,'jug001','jug002','2025-05-16 12:06:33','jug001'),
18 (7,'jug001','jug002','2025-05-16 12:09:38','jug001'),
19 (201,'JUG001','JUG004','2023-03-15 17:00:00','JUG004'),
20 (202,'JUG002','JUG003','2023-03-16 18:30:00','JUG003'),
21 (203,'JUG001','JUG002','2023-03-17 16:00:00','JUG002'),
22 (204,'JUG003','JUG004','2023-03-18 19:00:00','JUG003');
23

```

| Result Grid |                |            |            |                       |
|-------------|----------------|------------|------------|-----------------------|
|             | PARTITION_NAME | TABLE_ROWS | TABLE_NAME | PARTITION_DESCRIPTION |
| ▶           | p2023          | 4          | partidas   | 2024                  |
|             | p2024          | 0          | partidas   | 2025                  |
|             | p2025          | 2          | partidas   | 2026                  |
|             | pmax           | 0          | partidas   | MAXVALUE              |

Este sistema de particiones básicamente lo que hace es crear una tabla llamada partidas que almacena información sobre partidas jugadas entre dos jugadores. La tabla está particionada por rango de año según el campo fecha partida, lo cual significa que los datos se dividen internamente según el año en que ocurrió la partida: los datos del 2023 van a la partición p2023, los del 2024 a p2024, los del 2025 a p2025, y cualquier fecha posterior a 2025 irá a pmax. Esta partición ayuda a mejorar el rendimiento en consultas por fecha, ya que solo se accede a la partición relevante en lugar de recorrer toda la tabla. Además, al insertar registros como los del ejemplo, cada partida se guarda automáticamente en la partición correspondiente según el año de la fecha.

## Consultas que aprovechan la partición

```

1 • SELECT * FROM partidas
2 WHERE fecha_partida BETWEEN '2023-01-01' AND '2023-12-31';
3
4 • SELECT * FROM partidas
5 WHERE fecha_partida BETWEEN '2025-01-01' AND '2025-12-31';

```

|   | id_partida | id_jugador_1 | id_jugador_2 | fecha_partida       | ganador |
|---|------------|--------------|--------------|---------------------|---------|
| ▶ | 1          | jug001       | jug002       | 2025-05-16 12:06:33 | jug001  |
| * | 7          | jug001       | jug002       | 2025-05-16 12:09:38 | jug001  |
| * | NULL       | NULL         | NULL         | NULL                | NULL    |

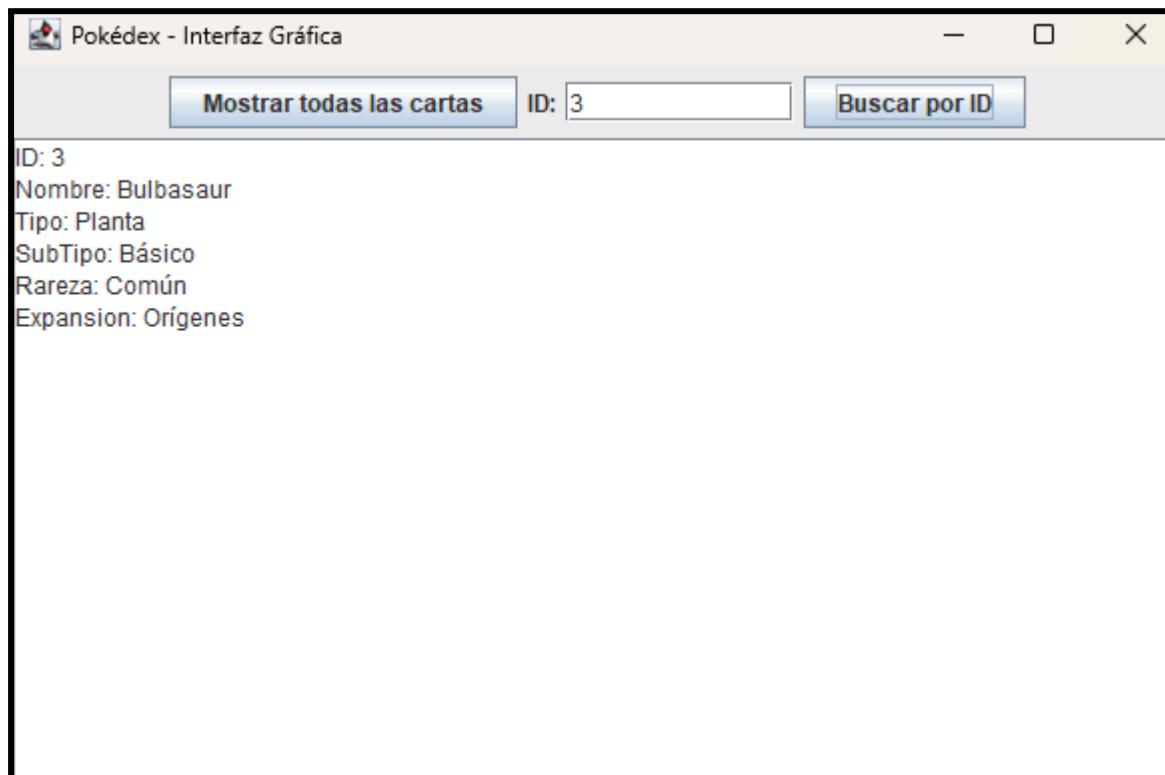
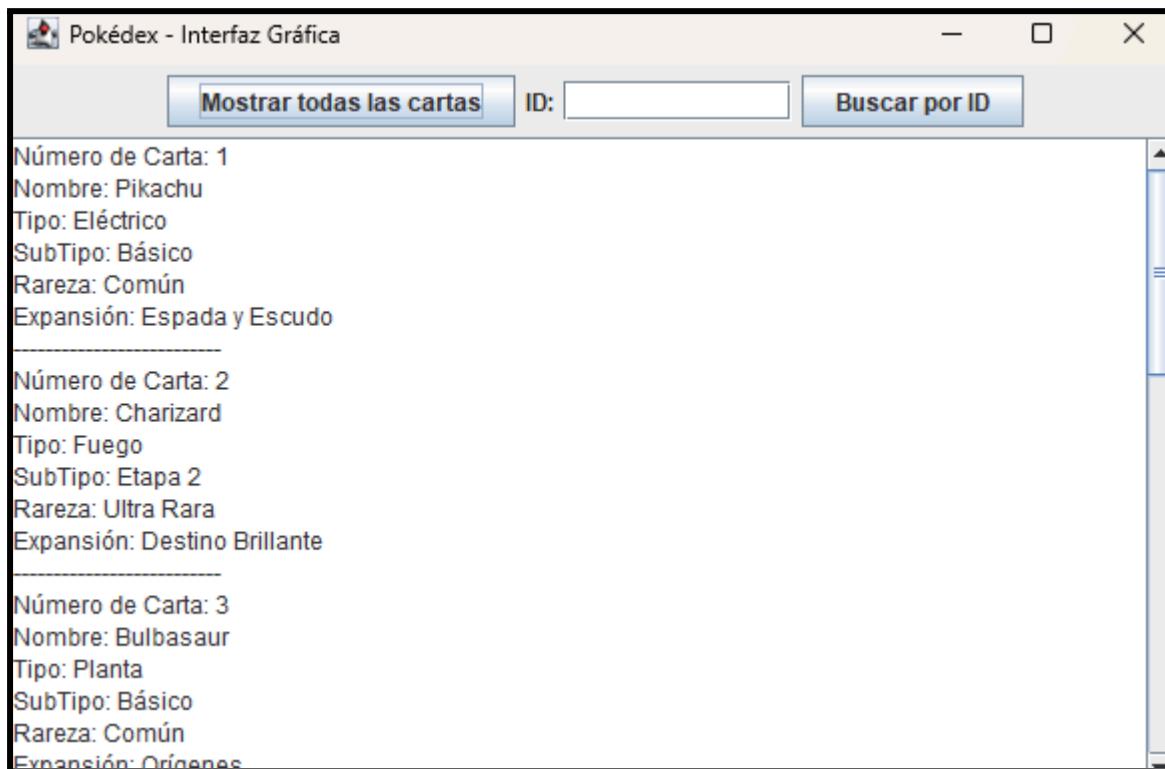
## Menú en Java para realizar consultas y modificaciones básicas en la base de datos (Código adjuntado)

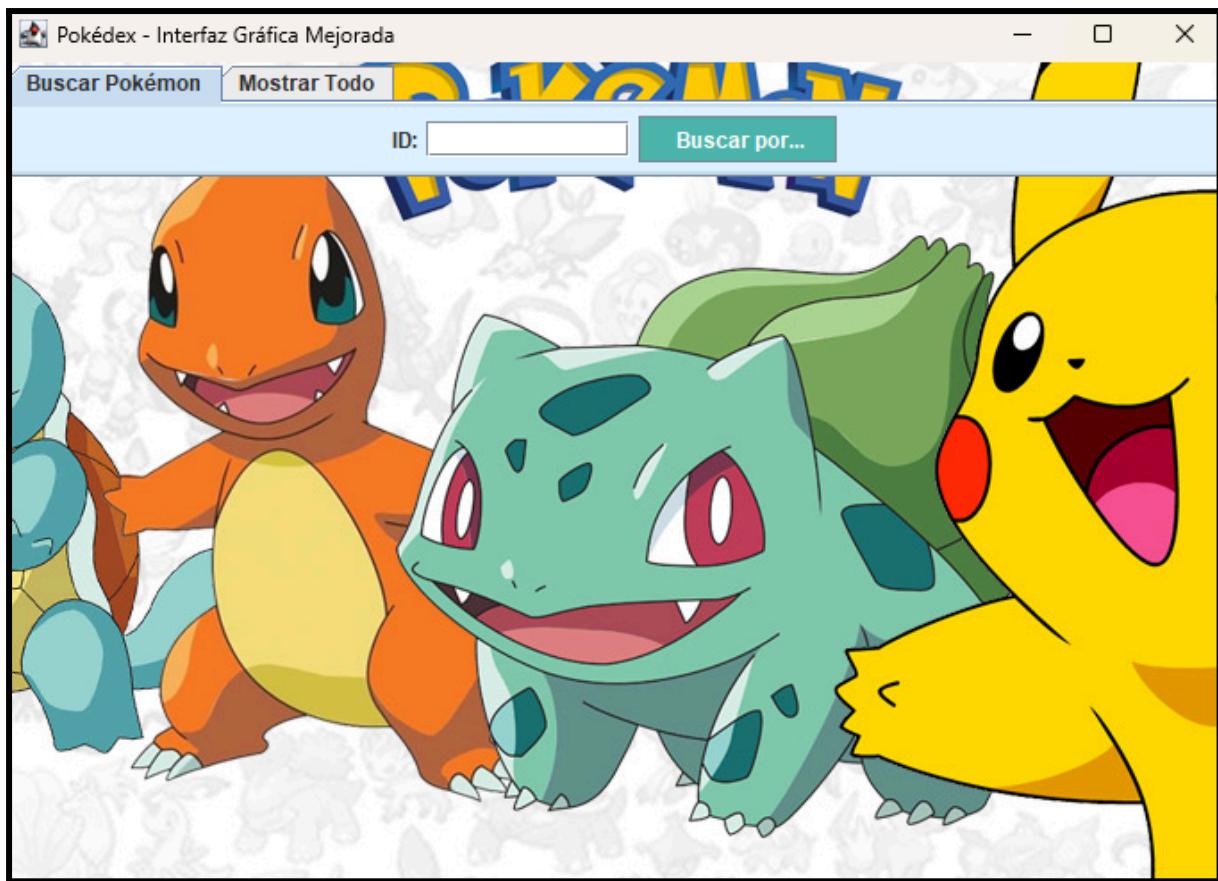
```

--- Menú ---
1. Visualizar todas las cartas
2. Visualizar una carta por su número
3. Filtrar cartas por tipo, subtipo o rareza
4. Añadir una nueva carta
5. Borrar una carta por su número
6. Actualizar una carta existente
0. Salir
Seleccione una opción:

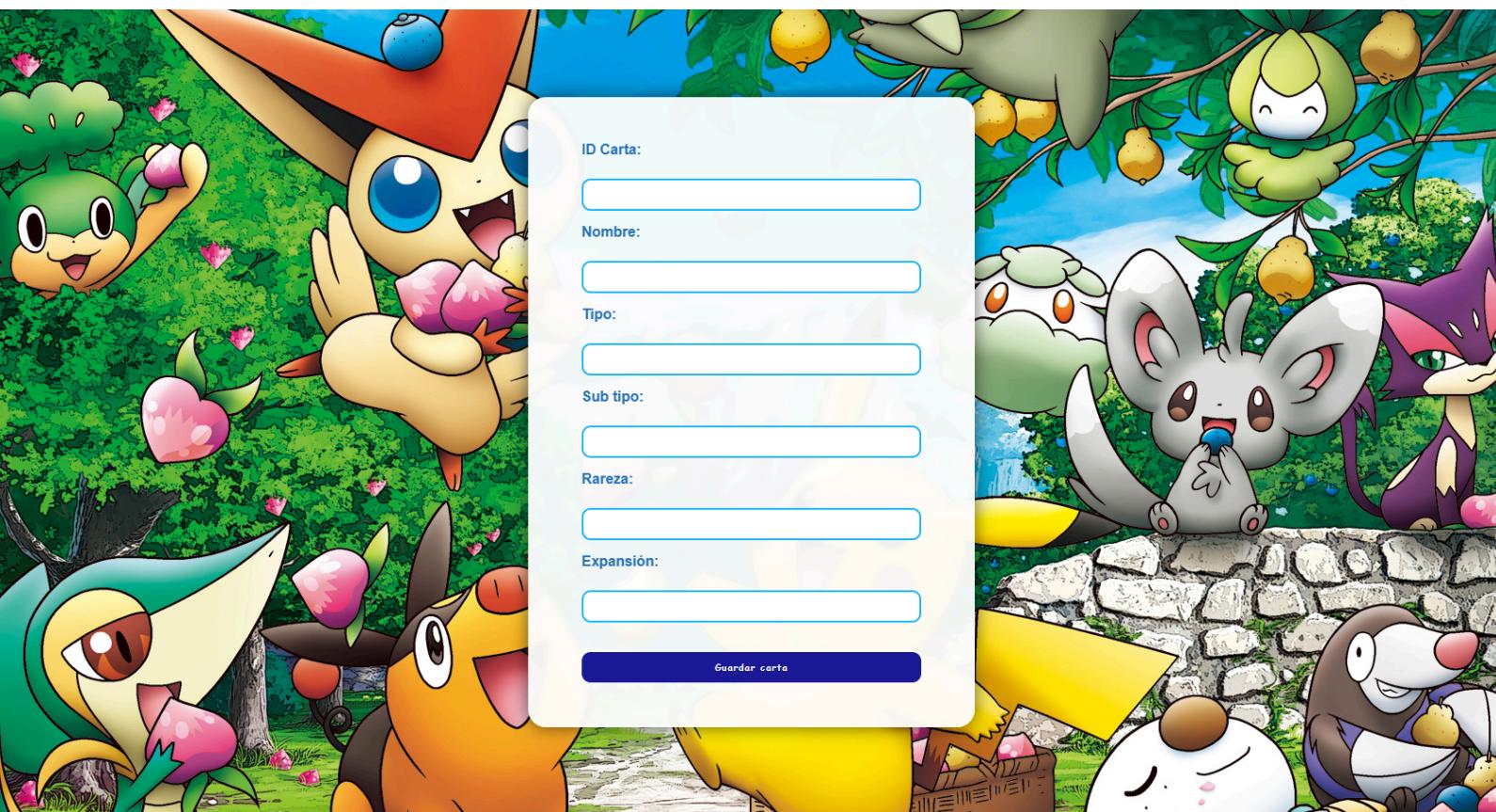
```

## Interfaz con Java (Código adjuntado)





## Entrada de datos desde app externa o formulario web



Carta guardada con éxito

ID Carta:

Nombre:

Tipo:

Sub tipo:

Rareza:

Expansión:

|   | id_carta | nombre     | tipo      | sub_tipo | rareza     | expansion_poke          |
|---|----------|------------|-----------|----------|------------|-------------------------|
| ▶ | 1        | Pikachu    | Eléctrico | Básico   | Común      | Espada y Escudo         |
|   | 2        | Charizard  | Fuego     | Etapa 2  | Ultra Rara | Destino Brillante       |
|   | 3        | Bulbasaur  | Planta    | Básico   | Común      | Orígenes                |
|   | 4        | Squirtle   | Agua      | Básico   | Común      | Orígenes                |
|   | 5        | Snorlax    | Normal    | Básico   | Rara       | Rebel Clash             |
|   | 6        | Gengar     | Fantasma  | Etapa 2  | Rara Holo  | Oscuridad Incandescente |
|   | 7        | Lucario    | Lucha     | Etapa 1  | Rara       | Evoluciones             |
|   | 8        | Eevee      | Normal    | Básico   | Común      | Truenos Perdidos        |
|   | 9        | vileplume  | veneno    | volador  | comun      | conversion dimens       |
|   | 10       | MegaGerman | Profe     | Acero    | Legendaria | atlantida               |
| * | NULL     | NULL       | NULL      | NULL     | NULL       | NULL                    |

## He Creado un proyecto con Node.js

1. Abrí una terminal (cmd) y cree una carpeta del proyecto:

```
mkdir mi-proyecto-cartas
cd mi-proyecto-cartas
```

2. He inicializado un proyecto Node.js:

```
npm init -y
```

3. Instale las dependencias necesarias:

```
npm install express mysql body-parser
```

---

Dentro de mi-proyecto-cartas, cree:

- index.js: servidor web con Express.
- db.js: conexión a MySQL.
- public/index.html: formulario HTML.
- package.json: se generó solo con npm init

## index.js: servidor web

```

1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const cors = require('cors');
4 const db = require('./db');
5
6 const app = express();
7 app.use(cors());
8 app.use(bodyParser.json());
9 app.use(express.static('public'));
10
11 // Ruta API
12 app.post('/api/cartas', (req, res) => {
13 const { id_carta, nombre, tipo, sub_tipo, rareza, expansion_poke } = req.body;
14
15 const sql = `INSERT INTO cartas (id_carta, nombre, tipo, sub_tipo, rareza, expansion_poke)
16 VALUES (?, ?, ?, ?, ?, ?)`;
17
18 db.query(sql, [id_carta, nombre, tipo, sub_tipo, rareza, expansion_poke], (err, result) => {
19 if (err) {
20 console.error('✗ Error al insertar carta:', err.message);
21 return res.status(500).send('Error al guardar la carta');
22 }
23 res.send('✓ Carta guardada con éxito');
24 });
25 });
26
27 app.listen(3000, () => {
28 console.log('Servidor corriendo en http://localhost:3000');
29 });

```

Este archivo:

- Lanza el servidor web.
- Atiende el formulario.
- Guarda los datos en la base MySQL.

## db.js: conexión a MySQL

```

1 // db.js
2 const mysql = require('mysql2');
3
4 const connection = mysql.createConnection({
5 host: 'localhost',
6 user: 'root',
7 password: '1234',
8 database: 'pokemon_tcg_pocket'
9 });
10
11 connection.connect((err) => {
12 if (err) {
13 console.error('✗ Error conectando a MySQL:', err.message);
14 return;
15 }
16 console.log('✓ Conectado a MySQL');
17 });
18
19 module.exports = connection;
20

```

Aquí se define cómo conectarse a mi base de datos MySQL.

1. En la terminal:

`node index.js`

2. En el navegador:

`http://localhost:3000`

3. Llenas el formulario y haces clic en "Guardar carta".

4. Los datos se envían a MySQL.

5. Si todo va bien, aparece:

 Carta guardada con éxito

---

Todo esto hace:

- Un formulario HTML se conecta con Node.js.
- Node.js recibe los datos.
- Los guarda en una base de datos MySQL.

```
Node.js v23.6.1
C:\Users\2DAM\mi-proyecto-cartas>node index.js
Servidor corriendo en http://localhost:3000
✓ Conectado a MySQL
```