¿Qué es un patrón de diseño?

Un **patrón de diseño** es una **solución reutilizable y probada** para resolver un problema común en el desarrollo de software, dentro de un contexto específico.

No es un código concreto que se copia y pega, sino **una guía o esquema** que describe cómo organizar las clases, objetos y responsabilidades para cumplir un objetivo de diseño de forma eficiente y mantenible.

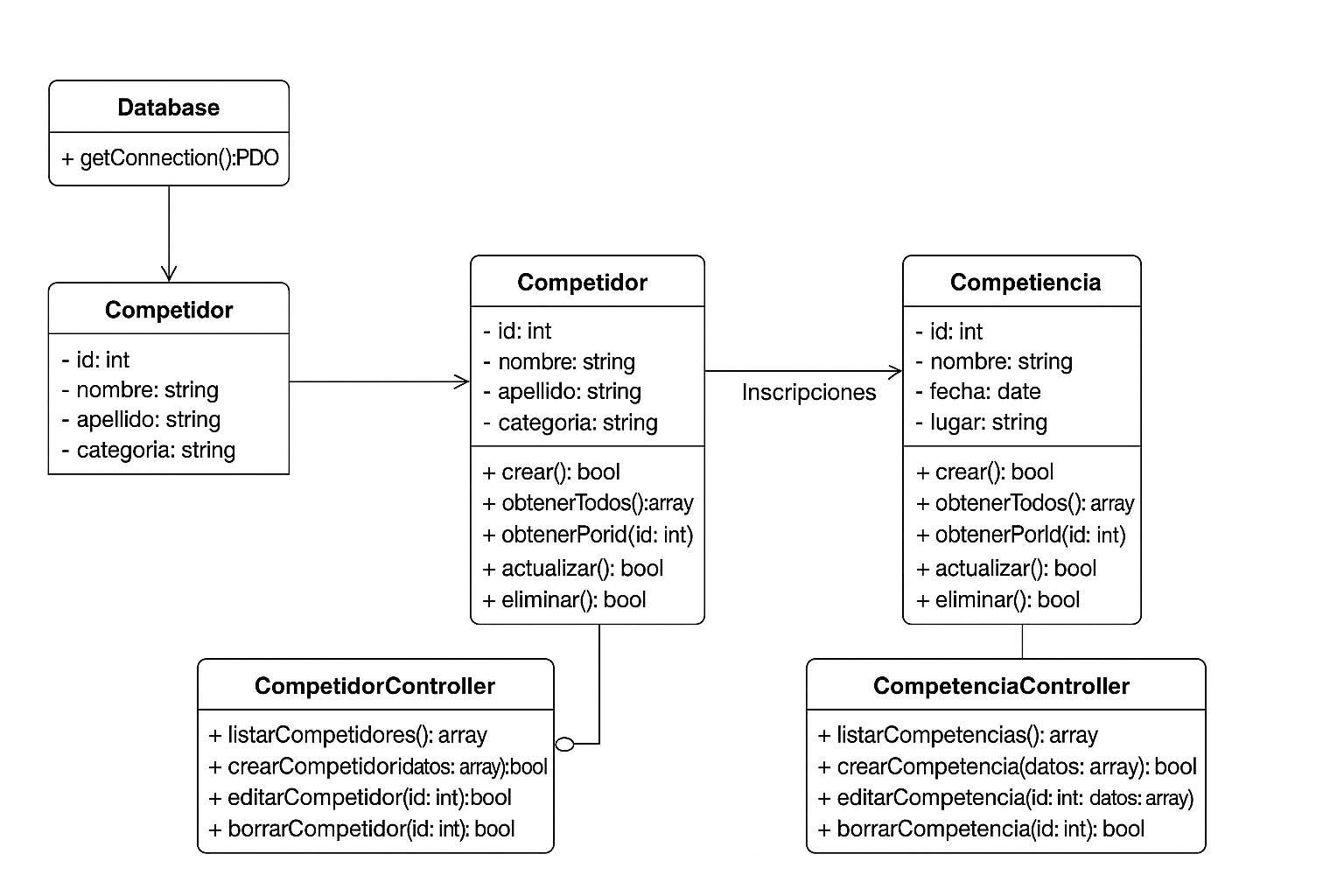
### **Características clave**

* **Reutilizable:** Se puede aplicar en distintos proyectos y situaciones similares.
* **Probado:** Ya fue utilizado y validado por otros desarrolladores.
* **Estándar de comunicación:** Facilita que los equipos entiendan rápidamente la arquitectura cuando se menciona por su nombre (“vamos a usar MVC”, “hagamos un Singleton”).
* **No depende de un lenguaje específico:** Se puede implementar en cualquier lenguaje de programación.

Que es MVC

El patrón de diseño [Modelo-Vista-Controlador](https://www.google.com/search?cs=1&sca_esv=2e47c951dd57c6d7&sxsrf=AE3TifNlvslXXRjB-ORBPcxuyfmVIg3KmQ%3A1754932044699&q=Modelo-Vista-Controlador&sa=X&ved=2ahUKEwjqvt6on4OPAxVwlJUCHZIxOJ8QxccNegQIAxAB&mstk=AUtExfCDcwWNk6DD2xpCxjB741fkPPpgwgOta2PyfvQpKae6GOj1imenyeJzTbpO3bXTQeQzrRrZwFvyOLcShaNcbxTqjsmz6_oIoeCGg56K1FNL-X3W-ha5Zt9ZYaDKzNGQ-aQ-qQg-RAIv2kqNKwyDvW_m3YpazfemmR_f1NBcTp-SJeU&csui=3) (MVC) es una arquitectura de software que separa la aplicación en tres componentes principales: Modelo, Vista y Controlador.

Realidad club de esgrima



club-esgrima/

│

├── backend/

│ ├── config/

│ │ └── Database.php

│ ├── models/

│ │ ├── Competidor.php

│ │ └── Competencia.php

│ ├── controllers/

│ │ ├── CompetidorController.php

│ │ └── CompetenciaController.php

│ ├── api/

│ │ ├── competidores.php

│ │ └── competencias.php

│ └── index.php

│

└── frontend/

├── index.html

└── js/

└── app.js

Modelo (models/)

**ué es:** Contiene las clases que representan la *lógica de negocio* y el acceso a datos.  
 Se encarga de interactuar con la base de datos y devolver la información procesada.

**En tu proyecto:**

* Competidor.php → Contiene métodos como getAllCompetidores(), addCompetidor(), deleteCompetidor(), etc.
* Competencia.php → Maneja las competencias, inscripción de competidores, ver quién ganó, etc.

**Responsabilidad:**

* No sabe nada de HTML ni de cómo se ve la página.
* Solo trabaja con datos: consultas SQL, validaciones, cálculos.

Controlador (controllers/)

**Qué es:** Actúa como un *intermediario* entre la Vista y el Modelo.  
 Recibe peticiones (por ejemplo, “agregar un competidor”), llama al Modelo para obtener o modificar datos y decide qué respuesta enviar.

**En tu proyecto:**

* CompetidorController.php → Llama al modelo Competidor.php para agregar, editar o listar competidores.
* CompetenciaController.php → Llama al modelo Competencia.php para manejar las inscripciones, resultados, etc.

**Responsabilidad:**

* Recibir solicitudes del frontend.
* Llamar a los métodos del modelo.
* Enviar la respuesta (normalmente en JSON si es API).

## **Vista (frontend/)**

* **Qué es:** Se encarga de la *presentación*, o sea, cómo se ve todo para el usuario.  
   Puede ser HTML, CSS, JavaScript, Bootstrap, etc.
* **En tu proyecto:**
  + index.html → Muestra el menú para agregar competidores y competencias.
  + js/app.js → Hace las peticiones AJAX a la API del backend.
  + Usás **Bootstrap** para darle estilo a formularios, botones, tablas.
* **Responsabilidad:**
  + Mostrar datos al usuario (por ejemplo, lista de competidores).
  + Enviar datos al controlador (por ejemplo, formulario de inscripción).

## **API (api/)**

* **Qué es:** Son endpoints que exponen las funciones del backend para que el frontend pueda comunicarse vía HTTP (AJAX/Fetch).
* **En tu proyecto:**
  + api/competidores.php → Recibe solicitudes relacionadas con competidores.
  + api/competencias.php → Recibe solicitudes relacionadas con competencias.

## **Configuración (config/)**

* **Qué es:** Contiene archivos de configuración que son usados por los modelos y controladores.  
   No forma parte estricta del MVC, pero es común tenerlo separado.
* **En tu proyecto:**
  + Database.php → Clase para conectarse a la base de datos con PDO o MySQLi.

Competencia.php

<?php

class Competencia {

private $conn;

private $table\_name = "competencias";

public $id;

public $competidor\_id;

public $nombre;

public $fecha;

public $gano;

public $tipo;

public function \_\_construct($db) {

$this->conn = $db;

}

public function read() {

$query = "SELECT c.\*,

CONCAT(co.nombre, ' ', co.apellido) AS competidor

FROM " . $this->table\_name . " c

INNER JOIN competidores co ON co.id = c.competidor\_id

ORDER BY fecha DESC";

$stmt = $this->conn->prepare($query);

$stmt->execute();

return $stmt;

}

public function create() {

$query = "INSERT INTO " . $this->table\_name . "

(competidor\_id, nombre, fecha, gano, tipo)

VALUES (:competidor\_id, :nombre, :fecha, :gano, :tipo)";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":competidor\_id", $this->competidor\_id);

$stmt->bindParam(":nombre", $this->nombre);

$stmt->bindParam(":fecha", $this->fecha);

$stmt->bindParam(":gano", $this->gano, PDO::PARAM\_BOOL);

$stmt->bindParam(":tipo", $this->tipo);

return $stmt->execute();

}

public function update() {

$query = "UPDATE " . $this->table\_name . "

SET competidor\_id=:competidor\_id, nombre=:nombre, fecha=:fecha,

gano=:gano, tipo=:tipo

WHERE id=:id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":competidor\_id", $this->competidor\_id);

$stmt->bindParam(":nombre", $this->nombre);

$stmt->bindParam(":fecha", $this->fecha);

$stmt->bindParam(":gano", $this->gano, PDO::PARAM\_BOOL);

$stmt->bindParam(":tipo", $this->tipo);

$stmt->bindParam(":id", $this->id);

return $stmt->execute();

}

public function delete() {

$query = "DELETE FROM " . $this->table\_name . " WHERE id=:id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":id", $this->id);

return $stmt->execute();

}

}

Competidor.php

<?php

class Competidor {

private $conn;

private $table\_name = "competidores";

public $id;

public $nombre;

public $apellido;

public $vi;

public $fechaNacimiento;

public $categoria;

public $arma;

public $mano;

public function \_\_construct($db) {

$this->conn = $db;

}

public function read() {

$query = "SELECT \* FROM " . $this->table\_name;

$stmt = $this->conn->prepare($query);

$stmt->execute();

return $stmt;

}

public function create() {

$query = "INSERT INTO " . $this->table\_name . "

(nombre, apellido, vi, fechaNacimiento, categoria, arma, mano)

VALUES (:nombre, :apellido, :vi, :fechaNacimiento, :categoria, :arma, :mano)";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":nombre", $this->nombre);

$stmt->bindParam(":apellido", $this->apellido);

$stmt->bindParam(":vi", $this->vi);

$stmt->bindParam(":fechaNacimiento", $this->fechaNacimiento);

$stmt->bindParam(":categoria", $this->categoria);

$stmt->bindParam(":arma", $this->arma);

$stmt->bindParam(":mano", $this->mano);

return $stmt->execute();

}

public function update() {

$query = "UPDATE " . $this->table\_name . "

SET nombre=:nombre, apellido=:apellido, vi=:vi, fechaNacimiento=:fechaNacimiento,

categoria=:categoria, arma=:arma, mano=:mano

WHERE id=:id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":nombre", $this->nombre);

$stmt->bindParam(":apellido", $this->apellido);

$stmt->bindParam(":vi", $this->vi);

$stmt->bindParam(":fechaNacimiento", $this->fechaNacimiento);

$stmt->bindParam(":categoria", $this->categoria);

$stmt->bindParam(":arma", $this->arma);

$stmt->bindParam(":mano", $this->mano);

$stmt->bindParam(":id", $this->id);

return $stmt->execute();

}

public function delete() {

$query = "DELETE FROM " . $this->table\_name . " WHERE id=:id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(":id", $this->id);

return $stmt->execute();

}

}

api

competencia.php

<?php

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Origin: \*");

header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE");

header("Access-Control-Allow-Headers: Content-Type");

require\_once "../config/Database.php";

require\_once "../models/Competencia.php";

$database = new Database();

$db = $database->getConnection();

$competencia = new Competencia($db);

$method = $\_SERVER["REQUEST\_METHOD"];

switch ($method) {

case "GET":

$stmt = $competencia->read();

$data = $stmt->fetchAll(PDO::FETCH\_ASSOC);

foreach ($data as &$row) {

$row['resultado'] = $row['gano'] ? 'Ganó' : 'Perdió';

}

echo json\_encode($data);

break;

case "POST":

$input = json\_decode(file\_get\_contents("php://input"));

$competencia->competidor\_id = $input->competidor\_id;

$competencia->nombre = $input->nombre;

$competencia->fecha = $input->fecha;

$competencia->gano = $input->gano;

$competencia->tipo = $input->tipo;

echo json\_encode(["success" => $competencia->create()]);

break;

case "PUT":

$input = json\_decode(file\_get\_contents("php://input"));

$competencia->id = $input->id;

$competencia->competidor\_id = $input->competidor\_id;

$competencia->nombre = $input->nombre;

$competencia->fecha = $input->fecha;

$competencia->gano = $input->gano;

$competencia->tipo = $input->tipo;

echo json\_encode(["success" => $competencia->update()]);

break;

case "DELETE":

$input = json\_decode(file\_get\_contents("php://input"));

$competencia->id = $input->id;

echo json\_encode(["success" => $competencia->delete()]);

break;

}