Primero vamos a crear **la entidad Alumno,** ya que es la información a mapear ya sea para listar o realizar una acción **contra la base de datos** (registrar/eliminar/actualizar). Esta clase contiene como atributos las columnas de la tabla.

Si queremos seguir el paradigma de la programación orientada a objetos teóricamente deberíamos tener una clase por cada tabla de la base de datos, que haga referencia a un objeto de la vida real, en este caso el objeto que crearíamos seria "Alumnos" y el alumno tendría un nombre, un apellido, sexo, fecha_nacimiento, pues bien eso serían los atributos del objeto y tendríamos un **método get y set** por cada atributo que servirán para establecer el valor de las propiedades y para conseguir el valor de cada atributo.

alumno.entidad.php

alumno.entidad.php

```php
<?php
class Alumno // Columnas de la Tabla Alumnos
{   private $_id;
    private $_Nombre;
    private $_Apellido;
    private $_Sexo;
    private $_FechaNacimiento;

    public function set_id($valor){ $this->_id = $valor; }
    public function set_nombre($valor){ $this->_Nombre = $valor; }
    public function set_apellido($valor){ $this->_Apellido = $valor; }
    public function set_sexo($valor){ $this->_Sexo = $valor; }
    public function set_fecha($valor){ $this->_FechaNacimiento = $valor; }

    public function get_id(){ return $this->_id ; }
    public function get_nombre(){ return $this->_Nombre; }
    public function get_apellido(){ return $this->_Apellido; }
    public function get_sexo(){ return $this->_Sexo; }
    public function get_fecha(){ return $this->_FechaNacimiento; }
}
?>
```

Otra forma seria, crear los atributos **set** y **get** por cada atributo, y luego pasar al siguiente atributo.

```php
public function setIDusuario($ID){
    $this->IDusuario=$ID;
    }

public function getIDusuario(){
    return $this->IDusuario;
    }

public function setNombre($Nom){
    $this->Nombre=$Nom;
    }

public function getNombre(){
    return $this->Nombre;
    }
```

alumno.model.php

alumno.model.php

```php
<?php
require_once("conexion.php");

class AlumnoModel
{
    //atributos
    //private $con;
    private $pdo;

    //metodos

    public function __construct()
    {
     $con = New conexion(); //instancia de la clase conexion
     $this->pdo = $con->getConexion();  //guardo en pdo la conexion misma
    }
```

Método Listar: Muestra todos los alumnos guardados en la tabla de Alumnos. Guarda cada instancia de alumnos en el arreglo ***result***

```php
    public function Listar()// Trae los alumnos de la tabla Alumnos de la base de datos
    {
        try
        {
            $result = array();

            $stm = $this->pdo->prepare("SELECT * FROM alumnos"); //directiva de traer toda la tabla alumno
            $stm->execute(); //ejecuta la consulta

            foreach($stm->fetchAll(PDO::FETCH_OBJ) as $r) //recorre una lista de objetos alumno que lo guarda en la variable r
            {
                $alm = new Alumno(); //se crea una instancia de alumno

                $alm->set_id($r->id); //guarda en la isntancia alumno, el id del objeto recuperado de la lista de objetos
                $alm->set_nombre($r->Nombre);
                $alm->set_apellido($r->Apellido);
                $alm->set_sexo($r->Sexo);
                $alm->set_fecha($r->FechaNacimiento);

                $result[] = $alm; //guarda cada isntancia de alumno en el arreglo result
            }

            return $result; //devuelve un arreglo de objetos alumnos
        }
        catch(Exception $e)
        {
            die($e->getMessage());
        }
}
```

Método Obtener: Busca un alumno en particular, mediante el id

```php
    public function Obtener($id) //busca un objeto alumno segun un id
    {
        try
        {
            $stm = $this->pdo->prepare("SELECT * FROM alumnos WHERE id = ?"); //prepara la consulta
            $stm->execute(array($id)); //ejecuta la consulta y pasa por parametro el id a buscar

            $r = $stm->fetch(PDO::FETCH_OBJ); //guarda en r el objeto de la clase alumno

            $alm = new Alumno(); //crea un objeto alm, una instancia de la clase alumno

            $alm->set_id($r->id); // guarda en la instancia alm, el id del objeto de la clase alumno
            $alm->set_nombre($r->Nombre);// lo mismo para el resto de los datos
            $alm->set_apellido($r->Apellido);
            $alm->set_sexo($r->Sexo);
            $alm->set_fecha($r->FechaNacimiento);

            return $alm; // segun el id especificado, devuelve un objeto de la clase alumno
        } catch (Exception $e)
        {
            die($e->getMessage());
        }
    }
```

Método Eliminar: Elimina un alumno de la tabla Alumnos de la base de datos

```php
public function Eliminar($id) //elimina un objeto de la clase alumno segun un id // elimina un registro de la tabla
{
    try
    {
        $stm = $this->pdo->prepare("DELETE FROM alumnos WHERE id = ?");       // crea la consulta
        $stm->execute(array($id)); // ejecuta la consulta
    } catch (Exception $e)
    {
        die($e->getMessage());
    }
}
```

Método Actualizar: Actualiza los datos de un alumno especifico.

```php
public function Actualizar(Alumno $data) //actualiza un registro de la tabla con un dato de tipo clase alumno
{
    try
    {
      $sql = "UPDATE alumnos SET
                    Nombre          = ?,
                    Apellido        = ?,
                    Sexo            = ?,
                    FechaNacimiento = ?
                WHERE id = ?"; // crea la consulta

      $this->pdo->prepare($sql)
           ->execute(
            array(
                $data->get_nombre(),
                $data->get_apellido(),
                $data->get_sexo(),
                $data->get_fecha(),
                $data->get_id()
                )
            ); //ejecutla la consulta
    } catch (Exception $e)
    {
       die($e->getMessage());
    }
}
```

Método Registrar: Guarda los datos del alumno dentro de la base de datos.

```php
    public function Registrar(Alumno $data)
    {
        try
        {
        $sql = "INSERT INTO alumnos (Nombre,Apellido,Sexo,FechaNacimiento)
                VALUES (?, ?, ?, ?)";

        $this->pdo->prepare($sql)
             ->execute(
            array(
                $data->get_nombre(),
                $data->get_apellido(),
                $data->get_sexo(),
                $data->get_fecha()
              )
            );
        } catch (Exception $e)
        {
            die($e->getMessage());
        }
    }
}
?>
```

Los **"?"**, lo usamos como comodines para escapar parametros. De esta forma evitamos concatenar nuestra consulta sql dejandolo propenso a un ataque sql conocido como "SQL Injection". Estas consultas se les conoce como **prepared statement**.