**PHP/MODEL/MANAGER/DAO.CLASS.PHP**

<?php

class DAO

{

    public static function add($obj)

    {

        $db = DbConnect::getDb();

        $class = get\_class($obj);

        $colonnes = $class::getAttributes();

        $requ = "INSERT INTO " . $class . "(";

        $values = "";

        $bindValue = [];

        for ($i = 1; $i < count($colonnes); $i++) {

            $methode = "get" . ucfirst($colonnes[$i]);

            if ($obj->$methode() != null) {

                $requ .= $colonnes[$i] . ",";

                $values .= ":" . $colonnes[$i] . ",";

            }

        }

        $requ = substr($requ, 0, strlen($requ) - 1);

        $values = substr($values, 0, strlen($values) - 1);

        $requ .= ") VALUES (" . $values . ")";

        //var\_dump($requ);

        $q = $db->prepare($requ);

        for ($i = 1; $i < count($colonnes); $i++) {

            $methode = "get" . ucfirst($colonnes[$i]);

            if ($obj->$methode() != null)

                $q->bindValue(":" . $colonnes[$i], $obj->$methode());

        }

        return $q->execute();

    }

       public static function update($obj)

    {

        $db = DbConnect::getDb();

        $class = get\_class($obj);

        $colonnes = $class::getAttributes();

        $requ = "UPDATE " . $class . " SET ";

        for ($i = 1; $i < count($colonnes); $i++) {

            $requ .= $colonnes[$i] . "=:" . $colonnes[$i] . ",";

        }

        $requ = substr($requ, 0, strlen($requ) - 1);

        $requ .= " WHERE " . $colonnes[0] . "=:" . $colonnes[0];

        $q = $db->prepare($requ);

        for ($i = 0; $i < count($colonnes); $i++) {

            $methode = "get" . ucfirst($colonnes[$i]);

            $q->bindValue(":" . $colonnes[$i], $obj->$methode());

        }

        return $q->execute();    }

public static function delete($obj)

    {

        $db = DbConnect::getDb();

        $class = get\_class($obj);

        $colonnes = $class::getAttributes();

        $methode = "get" . ucfirst($colonnes[0]);

        return $db->query("DELETE FROM " . $class . " WHERE " . $colonnes[0] . " = " . $obj->$methode());

    }

/\*\*

     \* Méthode privée qui sera appelée par la méthode select

     \* @param array $tab => Tableau de noms de colonnes ou agrégats de la BDD pour plus de détail allez voir la doc sur select.

     \* @return string => compose la partie SELECT de la méthode select

     \*/

    private static function elementSelect($tab)

    {

        $temp = "SELECT ";

        foreach ($tab as $uneCol) {

            $temp .= $uneCol . ", ";

        }

        return substr($temp, 0, strlen($temp) - 2);

    }

    /\*\*

     \* Méthode privée qui sera appelée par la méthode select

     \* @param array $conditions => tableaux qui contient les conditions pour plus de détail allez voir la doc sur select.

     \* @return string => compose la partie WHERE de la méthode select

     \*/

    private static function conditionSelect($conditions,$table)

    {

        $req = " WHERE ";

        foreach ($conditions as $nomColonne => $valeur) {

            if ($nomColonne != "fullTexte") {

                // cas du in

                if (is\_array($valeur)) {

                    $req .= $nomColonne . " IN (" . implode(",", $valeur) . ") AND ";

                } else if (!(strpos($valeur, "%") === false)) { //cas like

                    $req .= $nomColonne . ' LIKE "' . $valeur . '" AND ';

                } else if (strpos($valeur, "->")) { //cas between

                    $tab = explode("->", $valeur);

                    $req .= $nomColonne . " BETWEEN " . $tab[0] . " AND " . $tab[1] . " AND ";

                } else { //cas valeur simple

                    $req .= $nomColonne . " = \"" . $valeur . "\" AND ";

                }

            } elseif($valeur!="") {

                $rechercheFullTexte="";

                $colonnes  = $table::getAttributes();

                foreach ($colonnes as $col) {

                    if(strtolower(substr($col,0,4))!="date")

                    $rechercheFullTexte .= $col .' like "%'.$valeur.'%" OR ';

                }

                $rechercheFullTexte = substr($rechercheFullTexte, 0, strlen( $rechercheFullTexte ) - 3);

                $req .= "(".$rechercheFullTexte.") AND ";

            }

        }

        //On retire le dernier AND

        $req = substr($req, 0, strlen($req) - 4);

        //var\_dump($req);

        return $req;

    }

}

    /\*\*

     \*

     \* @param array $nomColonnes => contient le noms des champs désirés dans la requête.

     \* Exemple :  [nomColonne1,nomColonne2] => "SELECT nomColonne1, nomColonne2"

     \*

     \* @param string $table => contient Nom de la table sur laquelle la requête sera effectuée.

     \* Exemple : nomTable => "FROM nomTable"

     \*

     \* @param array $conditions => null par défaut, attendu un tableau associatif

     \* qui peut prendre plusieurs formes en fonction de la complexité des conditions.

     \*  Exemples : tableau associatif

     \*  [nomColonne => '1'] => "WHERE nomColonne = 1"

     \*  [nomColonne => ['1','3']] => "WHERE nomColonne in (1,3)"

     \*  [nomColonne => '%abcd%'] => "WHERE nomColonne like "abcd" "

     \*  [nomColonne => '1->5'] => "WHERE nomColonne BETWEEN 1 and 5 "

     \*  Si il y a plusieurs conditions alors :

     \*  [nomColonne1 => '1', nomColonne2 => '%abcd%' ] => "WHERE nomColonne1 = 1 AND nomColonne2 LIKE "%abcd%"

     \*  [fullTexte]=>'test'=> "WHERE nomColonne1 like "%test%" OR nomColonne2 LIKE "%test%"

     \*

     \* @param string $orderBy => null par défaut, contient un string qui contient les noms de colonnes et le type de tri

     \* Exemple :"nomColonne1 , nomColonne2 DESC" => "Order By nomColonne1 , nomColonne2 DESC"

     \*

     \* @param string $limit  => null par défaut, contient un string pour donner la délimitations des enregistrements de la BDD

     \* Exemples :

     \* "1" => "LIMIT 1"

     \* "1,2"=> "LIMIT 1,2"

     \*

     \* @param boolean $api => false par défaut, mettre true si on souhaite recevoir la réponse sous forme de string sinon sous forme d'objets.

     \*

     \* @param bool $debug => contient faux par défaut mais s'il on le met a vrai, on affiche la requete qui est effectuée.

     \*

     \* @return [array ou object] $liste => résultat de la requête revoie false si la requête s'est mal passé sinon renvoie un tableau.

     \*/

    public static function select(?array $nomColonnes=null, string $table, array $conditions = null, string $orderBy = null, string $limit = null, bool $api = false, bool $debug = false)

    {

        $db = DbConnect::getDb();

        $nomColonnes = ($nomColonnes == null) ? $table::getAttributes() : $nomColonnes;

        $string = json\_encode($nomColonnes) . $table . json\_encode($conditions) . $orderBy . $limit . $api . $debug;

        if (strpos($string, ";")) {

            return false;

        } else if (!empty($nomColonnes)  && ($table != null && $table != "")) {

            $cols = self::elementSelect($nomColonnes);

            $t = " FROM ". $table;

            if ($conditions != null) {

                $conditions = self::conditionSelect($conditions,$table);

            }

            if ($orderBy != null) {

                $orderBy = " ORDER BY " . $orderBy;

            }

            if ($limit != null) {

                $limit = " LIMIT " . $limit;

            }

            $q = $db->query($cols . $t . $conditions . $orderBy . $limit);

            if ($debug) // Si le debug est a true on affiche la requete qui est envoyée en base de données

            {

                var\_dump($cols . $t . $conditions . $orderBy . $limit);

            }

            $liste = [];

            if (!$q) return false;

            while ($donnees = $q->fetch(PDO::FETCH\_ASSOC)) { // on récupère les enregistrements de la BDD

                if ($donnees != false) {

                    if ($api) { // On vérifie si api est a true, on renvoie un string sinon des objets liés a à la table donnée en paramètres.

                        $liste[] = $donnees;

                    } else {

                        $liste[] = new $table($donnees);

                    }

                }

            }

            return $liste;

        }

        return false;

    }