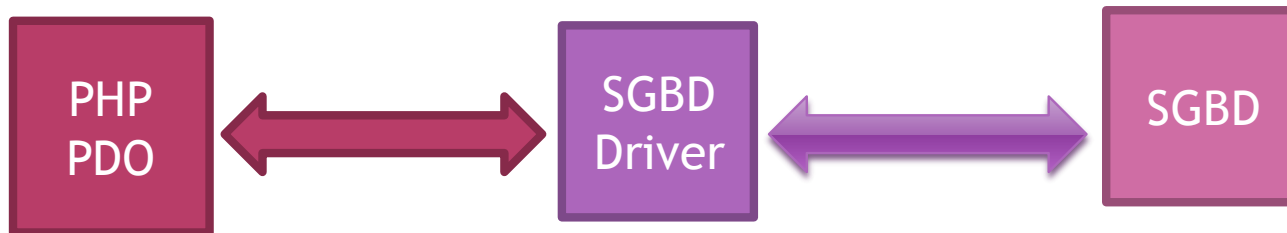


PHP - PDO

PHP DATA OBJECTS

PDO C'EST QUOI ?

- PDO définit une interface d'abstraction pour accéder de manière uniforme aux bases de données.
- Il existe des implémentations pour la plupart des SGBD connus.



SE CONNECTER A MYSQL

⦿ Avec les fonction mysql_*

```
<?php
```

```
$link = mysql_connect('localhost', 'user', 'pass');  
mysql_select_db('testdb', $link);  
mysql_set_charset('UTF-8', $link);
```

⦿ Avec PDO

```
<?php
```

```
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
             'username',  
             'password');
```

Le premier paramètre est la DSN (Data Source Name). Il définit le driver du SGBD auquel on veut accéder.

Les deux autres sont le nom d'utilisateur et le mot de passe

On peut passer un troisième qui contient un tableau de paramètres

EXECUTER DES REQUÊTES SIMPLES

```
<?php
```

```
    $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
                 'username', 'password');
```

```
// Requête SELECT :
```

```
    $stmt = $db->query("SELECT * FROM table");
```

```
// $stmt est un objet de type « PDOStatement »
```

```
// Nombre d'enregistrements
```

```
    echo $stmt->rowCount();
```

```
// Ici, on récupère tout les enregistrements dans un tableau associatif
```

```
    $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
// PDO présente plusieurs constantes accessibles par PDO::*
```

EXECUTER DES REQUÊTES SIMPLES

```
<?php
```

```
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
             'username', 'password');
```

```
// Requête INSERT:
```

```
$ret = $db->exec("INSERT INTO table (champ) VALUES ('valeur')");
```

```
// Récupération de la clé de l'enregistrement ajouté
```

```
$id = $db->lastInsertId();
```

EXECUTER DES REQUÊTES SIMPLES

```
<?php
```

```
    $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
                 'username', 'password');
```

```
// Requête DELETE :
```

```
    $ret = $db->exec("DELETE FROM table (champ) WHERE id<10");
```

```
// Requête UPDATE :
```

```
    $ret = $db->exec("UPDATE table SET champ='valeur' WHERE id=10");
```

```
// $ret représente le nombre d'enregistrements affectés
```

LES REQUÊTES PRÉPARÉES

```
<?php
```

```
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
             'username', 'password');
```

```
// Requête préparée : $stmt est de type « PDOStatement
```

```
$id = 20;
```

```
stmt = $db->prepare("DELETE FROM table (champ) WHERE id=?");
```

```
// Faire correspondre le paramètre représenté par « ? » :
```

```
$stmt->bindValue(1, $id, PDO::PARAM_INT);
```

```
// Exécuter la requête :
```

```
$stmt->execute();
```

```
// $stmt->rowCount() représente le nombre d'enregistrements affectés
```

localhost/test1/index.php

Ajouter un utilisateur

Login

Password

ID	LOGIN	PASSWORD	
1	toto	toto	<input type="button" value="Supprimer"/> <input type="button" value="Editer"/>

```

<?php
$login = $_POST['log'];
$pass = $_POST['pass'];

$dns = "mysql:host=localhost;dbname=ehtp";
$user = "root";
$password = "";

$pdo = new PDO($dns, $user, $password);

$sql = "INSERT INTO user VALUES ('', '$login', '$pass')";

$ret = $pdo->exec($sql);

header("location:index.php");
  
```

localhost/test/edit.php?idd=1

Editer l'utilisateur

```

<?php
$idd = $_GET['idd'];

$dns = "mysql:host=localhost;dbname=ehtp";
$user = "root";
$password = "";

$pdo = new PDO($dns, $user, $password);

$sql = "DELETE FROM user WHERE id=$idd";

$ret = $pdo->exec($sql);

header("location:index.php");
  
```

```

<?php
$id = $_POST['idd'];
$login = $_POST['log'];
$pass = $_POST['pass'];

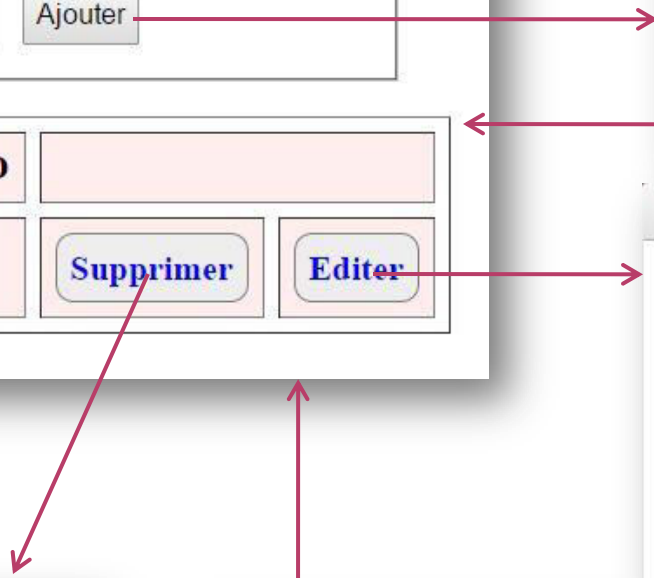
$dns = "mysql:host=localhost;dbname=ehtp";
$user = "root";
$password = "";

$pdo = new PDO($dns, $user, $password);

$sql = "UPDATE user SET login='$login',password='$pass' WHERE id=$id";

$ret = $pdo->exec($sql);

header("location:index.php");
  
```



LES REQUÊTES PREAPARÉES

```
<?php
```

```
    $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
                 'username', 'password');
```

```
// Requête préparée : $stmt est de type « PDOStatement
```

```
    stmt = $db->prepare("SELECT * FROM table WHERE nom LIKE ?");
```

```
// Faire correspondre le paramètre représenté par « ? » :
```

```
    $stmt->bindValue(1, '%.$nom.%', PDO::PARAM_STR);
```

```
// Exécuter la requête :
```

```
    $stmt->execute();
```

```
    $rows = $stmt->fetchAll(PDO::FETCH_OBJ);
```

```
// Ici, $rows représente les enregistrements retournés sous forme  
d'objets
```

LES REQUÊTES PREAPARÉES

```
<?php
```

```
    $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
                 'username', 'password');
```

```
// Requête préparée : $stmt est de type « PDOStatement
```

```
    stmt = $db->prepare("SELECT * FROM table WHERE nom = :nom");
```

```
// Faire correspondre le paramètre représenté par « ? » :
```

```
    $stmt->bindValue('nom', 'momo', PDO::PARAM_STR);
```

```
// Exécuter la requête :
```

```
    $stmt->execute();
```

```
    $rows = $stmt->fetchAll(PDO::FETCH_OBJ);
```

```
// Ici, $rows représente les enregistrements retournés sous forme  
d'objets
```

GERER LES EXCEPTIONS

```
<?php
```

```
try {
```

```
    $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',  
                 'username', 'password');
```

```
    $stmt = $db->prepare("SELECT * FROM table WHERE nom = :nom");
```

```
    $stmt->bindValue('nom', 'momo', PDO::PARAM_STRING);
```

```
    $stmt->execute();
```

```
    $rows = $stmt->fetchAll(PDO::FETCH_OBJ);
```

```
} catch( PDOException $e){
```

```
    echo "Une erreur s'est produite. " . $e->getMessage();
```

```
}
```

GERER LES TRANSACTIONS

```
<?php
try {
    $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8',
                'username', 'password');

    $db->beginTransaction(); // Démarrer une transaction

    stmt = $db->prepare("INSERT INTO COMMANDES ... ");
    $stmt->execute(...);

    $stmt = $db->prepare("INSERT INTO CLIENTS... ");
    $stmt->execute(...);

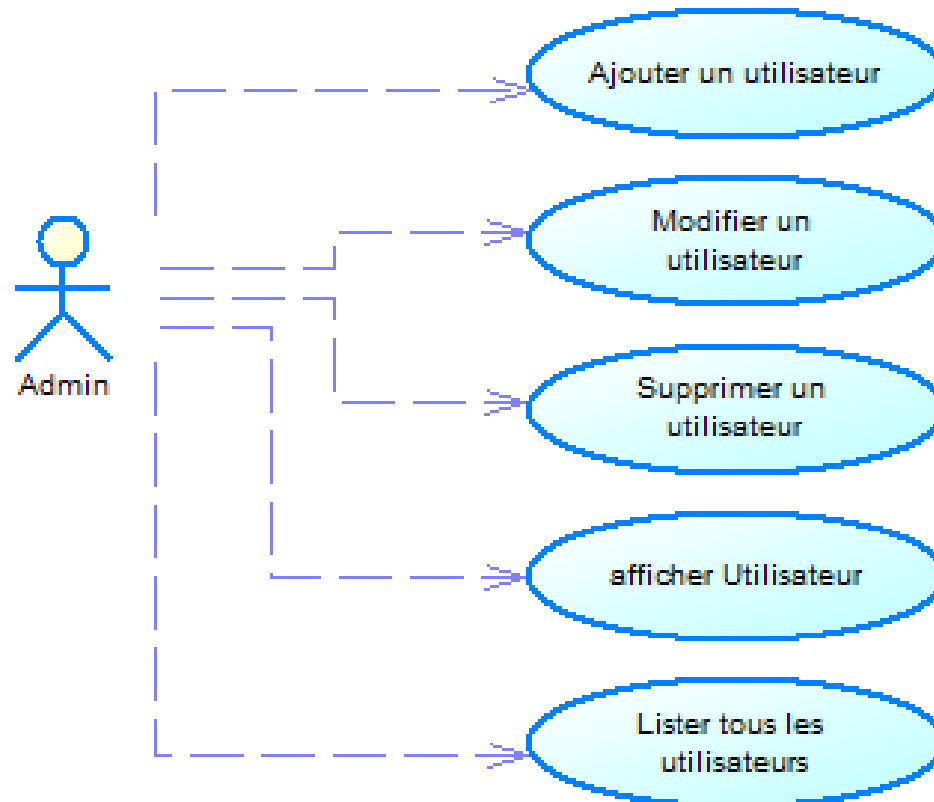
    $db->commit(); // Valider la transaction

} catch( PDOException $e){
    echo "Une erreur s'est produite. " . $e->getMessage();
    $db->rollBack(); // Annuler la transaction
}
```

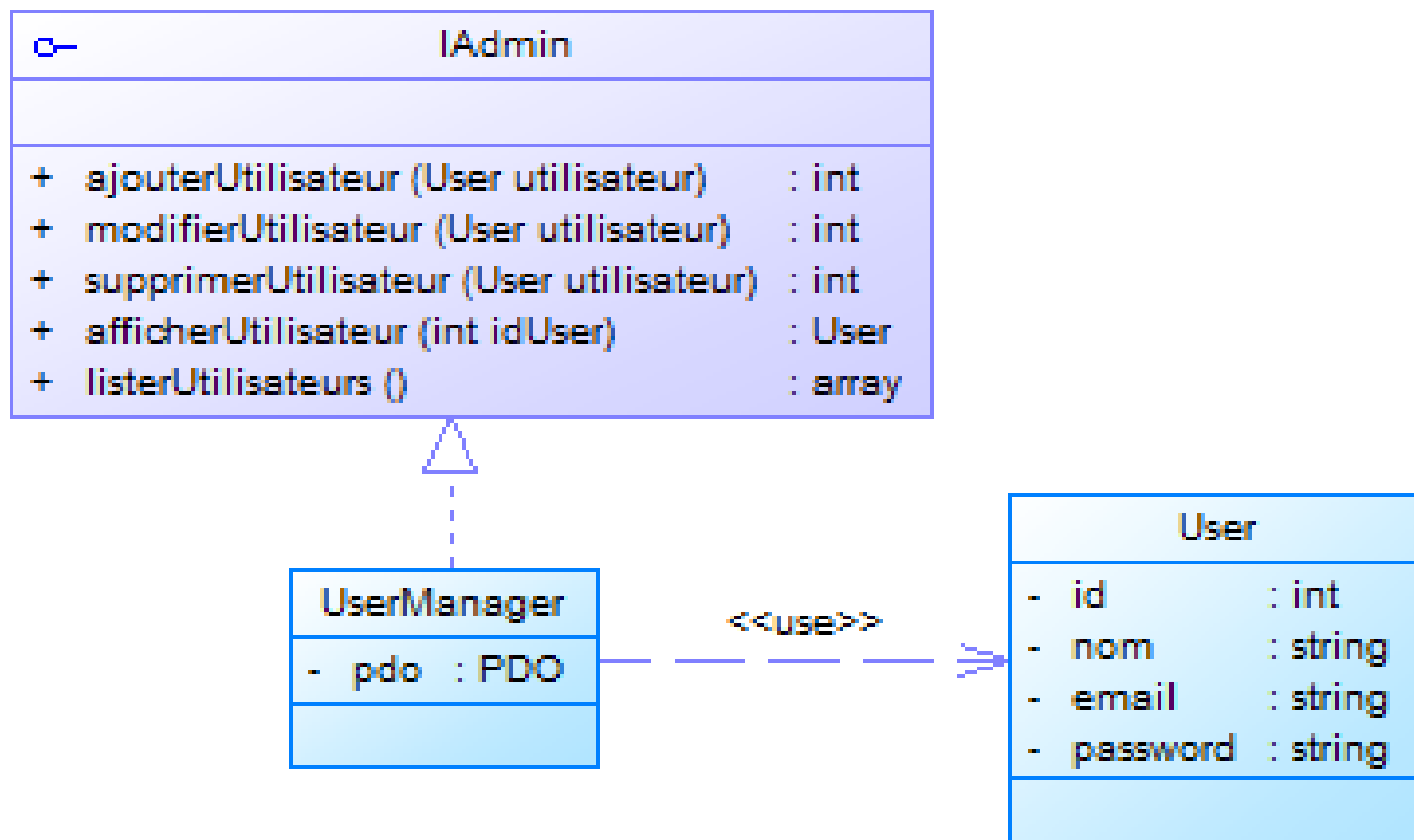
APPLICATION

- On se propose de réaliser une application pour gérer les utilisateurs.
- Un utilisateur est caractérisé par un identificateur (entier), un nom (string), un email (string) et un mot de passe.
- L'application doit permettre de :
 - ajouter un utilisateur
 - modifier un utilisateur
 - Supprimer un utilisateur
 - Afficher un utilisateur
 - Afficher tout les utilisateurs

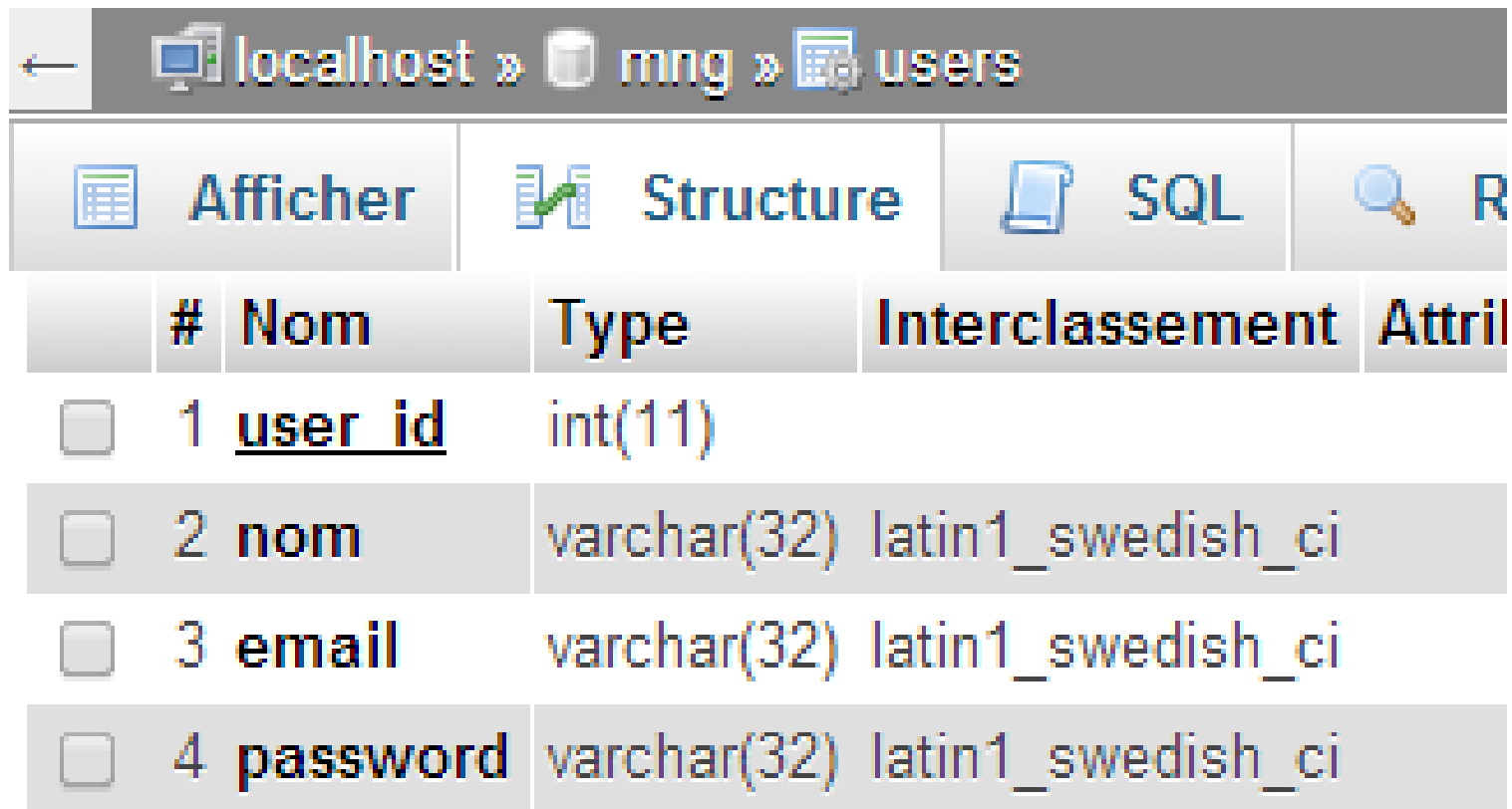
ANALYSE : USE CASES



ANALYSE : DIAGRAMME DE CLASSES



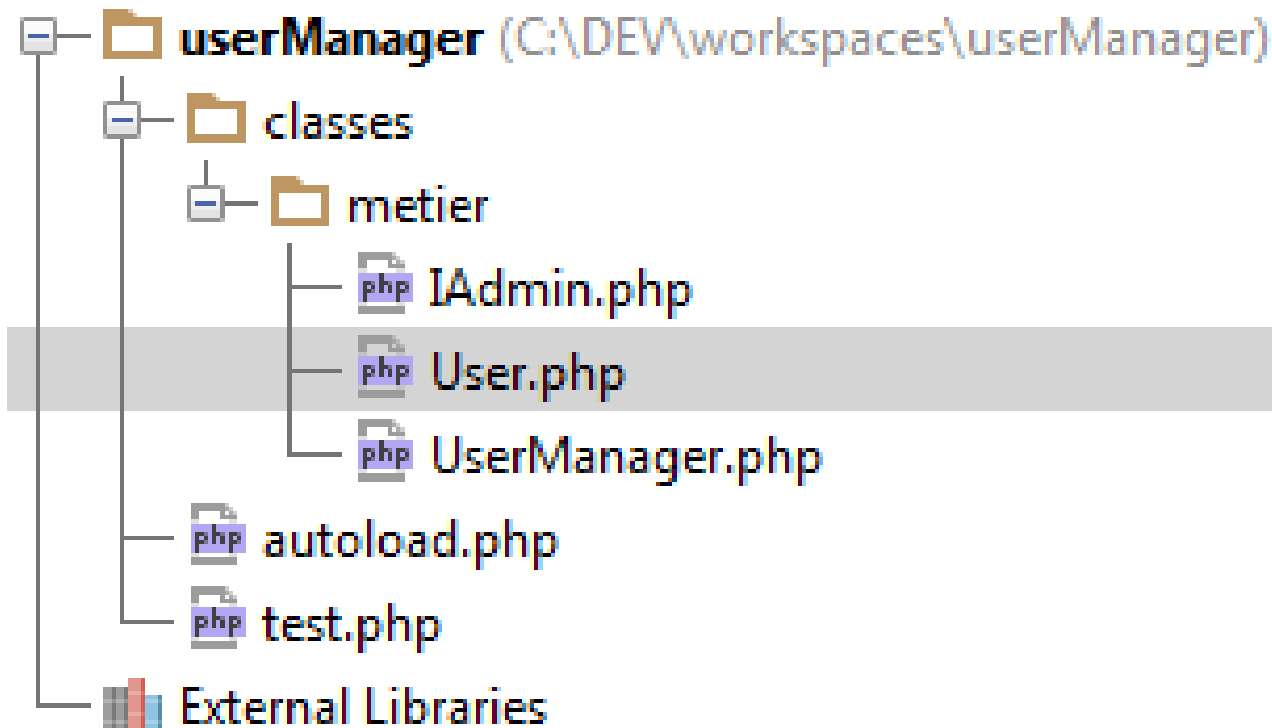
BASE DE DONNÉES



The screenshot shows a database management interface. The breadcrumb path is 'localhost » mng » users'. The 'Structure' tab is selected, displaying the table structure for 'users'. The table has four columns: 'user_id' (int(11)), 'nom' (varchar(32) latin1_swedish_ci), 'email' (varchar(32) latin1_swedish_ci), and 'password' (varchar(32) latin1_swedish_ci). The 'user_id' column is underlined, indicating it is the primary key.

#	Nom	Type	Interclassement	Attribut
<input type="checkbox"/> 1	<u>user_id</u>	int(11)		
<input type="checkbox"/> 2	nom	varchar(32)	latin1_swedish_ci	
<input type="checkbox"/> 3	email	varchar(32)	latin1_swedish_ci	
<input type="checkbox"/> 4	password	varchar(32)	latin1_swedish_ci	

STRUCTURE DU PROJET



IMPLEMENTATION : USER

```
// classes/metier/User.php
namespace metier;
class User {
    private $idUser;
    private $name;
    private $email;
    private $password;

    public function __construct($name, $email, $password)
    {
        $this->name = $name;
        $this->email = $email;
        $this->password = $password;
    }
// getters et setters
}
```

IMPLEMENTATION : IADMIN

```
// classes/metier/IAdmin.php
namespace metier;
interface IAdmin {

    function ajouterUtilisateur(User $user);

    function modifierUtilisateur(User $user);

    function supprimerUtilisateur(User $user);

    function getUtilisateur($idUser);

    function listerUtilisateurs();

}
```

IMPLEMENTATION : USERMANAGER

```
// classes/metier/UserManager.php
namespace metier;
use metier\User;
class UserManager implements IAdmin
{
    private $pdo = null;
    function __construct($dbname, $userName, $password)
    {
        $this->pdo = new \PDO(
            "mysql:host=localhost;dbname=$dbname;charset=utf8",
            $userName, $password);
        $this->pdo->setAttribute(
            \PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);
    }
    function ajouterUtilisateur(User $user)
    {
        $stmt = $this->pdo->prepare("INSERT INTO users VALUES(null,?,?,?)");
        $stmt->bindValue(1, $user->getName());
        $stmt->bindValue(2, $user->getEmail());
        $stmt->bindValue(3, $user->getPassword());
        $stmt->execute();
    }
}
```

IMPLEMENTATION : USERMANAGER

```
// Suite
```

```
function modifierUtilisateur(User $user)
{
    $stmt = $this->pdo->prepare(
        "UPDATE users SET nom=:nom, email=:email,
        password=:password WHERE user_id=:id");
    $stmt->bindValue('id', $user->getIdUser());
    $stmt->bindValue('nom', $user->getName());
    $stmt->bindValue('email', $user->getEmail());
    $stmt->bindValue('password', $user->getPassword());
    $stmt->execute();
}

function supprimerUtilisateur(User $user)
{
    $sql = "DELETE FROM users WHERE user_id=?";
    $stmt = $this->pdo->prepare($sql);
    $stmt->execute(array($user->getIdUser()));
}
```

IMPLEMENTATION : USERMANAGER

```
// Suite
```

```
function getUtilisateur($idUser)
{
    $sql = "SELECT * FROM users WHERE user_id=?";
    $stmt = $this->pdo->prepare($sql);
    $stmt->execute(array($idUser));
    $row = $stmt->fetch(\PDO::FETCH_OBJ);
    $user = null;
    if ($row) {
        $user = new User($row->nom, $row->email,
            $row->password);
        $user->setIdUser($row->user_id);
    }
    return $user;
}
```

IMPLEMENTATION : USERMANAGER

```
// Suite
```

```
function listerUtilisateurs()  
{  
    $sql = "SELECT * FROM users";  
    $stmt = $this->pdo->prepare($sql);  
    $stmt->execute();  
    $rows = $stmt->fetchAll(\PDO::FETCH_OBJ);  
    $users = array();  
    foreach($rows as $row){  
        $user = new User($row->nom, $row->email,  
                        $row->password);  
        $user->setIdUser($row->user_id);  
        $users[] = $user;  
    }  
    return $users;  
}
```

AUTOLOAD DES CLASSES :

```
// autoload.php
function __autoload($className) {
    try{
        $path = "classes" .
            DIRECTORY_SEPARATOR .
            str_replace("\\",
                DIRECTORY_SEPARATOR, $className) .
            ".php";

        require_once($path);

    }catch(Exception $e) {
        echo $e->getMessage();
    }
}
```


TESTS :

```
// test.php
use metier\User;
use metier\UserManager;

require_once "autoload.php";

// Création du manager
$um = new UserManager("mng", "root", "");

$user1 = new User("momo", "momo", "momo");
$user2 = new User("toto", "toto", "toto");

// Test Ajout des utilisateurs
echo "<hr>Ajout des utilisateurs<br>";
$um->ajouterUtilisateur($user1);
$um->ajouterUtilisateur($user2);
```

TESTS :

```
// Suite
// Test Lister Utilisateurs
echo "<hr>Lister les utilisateurs<br>";
foreach($um->listerUtilisateurs() as $user) {
    var_dump($user);
}

// Test Lister Utilisateurs
echo "<hr>Modifier tous les utilisateurs<br>";
foreach($um->listerUtilisateurs() as $user) {
    $user->setName(strtoupper($user->getName()));
    $um->modifierUtilisateur($user);
}
foreach($um->listerUtilisateurs() as $user) {
    var_dump($user);
}
```

TESTS :

```
// Suite
// Test Suppression Utilisateurs
echo "<hr>Suppression tous les utilisateurs<br>";
foreach($sum->listerUtilisateurs() as $user) {
    $sum->supprimerUtilisateur($user);
}

// Afficher s'il ya encore des utilisateurs
var_dump($sum->listerUtilisateurs());
```

TESTS : RÉSULTATS

Ajout des utilisateurs

Lister les utilisateurs

```
object(metier\User) [8]
  private 'idUser' => string '17' (length=2)
  private 'name' => string 'momo' (length=4)
  private 'email' => string 'momo' (length=4)
  private 'password' => string 'momo' (length=4)

object(metier\User) [9]
  private 'idUser' => string '18' (length=2)
  private 'name' => string 'toto' (length=4)
  private 'email' => string 'toto' (length=4)
  private 'password' => string 'toto' (length=4)
```

Modifier tous les utilisateurs

```
object(metier\User) [7]
  private 'idUser' => string '17' (length=2)
  private 'name' => string 'MOMO' (length=4)
  private 'email' => string 'momo' (length=4)
  private 'password' => string 'momo' (length=4)

object(metier\User) [8]
  private 'idUser' => string '18' (length=2)
  private 'name' => string 'TOTO' (length=4)
  private 'email' => string 'toto' (length=4)
  private 'password' => string 'toto' (length=4)
```

Suppression de tous les utilisateurs

```
array (size=0)
  empty
```