Construimos la clase Users con un constructor público que crea una conexión a la BD (con la extensión PDO) y algunos métodos para interactuar con la BD

users.php

```
<?php
class Users
{  private  $Idb;      private  $filas = array();
   public  function  __construct()
    { $host = 'localhost';      $db = 'database';      $usuario = 'user';      $password = 'password';
      $this->Idb = new PDO( "mysql:host=$host; dbname=$db" ,  $usuario ,  $password );
    }
   private function  set_names()            //método para evitar problemas de ñ y acentos con la base de datos
    {  return $this->Idb->query("SET NAMES 'utf8'");   }

   public  function  usuarios()             //método con el que obtenemos datos de la tabla usuarios
    {  self::set_names();
      $consulta = $this->Idb->prepare("SELECT * FROM  usuarios");
      $consulta->execute();
      if ($consulta->rowCount()>0) :
         while($row=$consulta->fetch())    {   $this->filas[] = $row;    }
         return $this->filas;
      endif;
    }
}
?>
```

Si queremos hacer uso de esta clase en cualquier archivo deberíamos hacer lo siguiente.

```
<?php
require_once    'users.php';

$user = new Users();    //crea  una instancia y dentro tenemos  todas las propiedades y métodos de la clase

$datos = $user->usuarios();  //  llamamos   al método usuarios y mostramos los datos

echo “<table>”;
foreach  ( $datos as $fila)
 { echo  “<tr>”;
   foreach  ( $fila    as  $columna)
     { echo  “ <td>  $columna </td> “; }
   echo  “</tr>”;
  }
echo “</table>”;

?>
```

Esta es la forma normal de acceder a una clase desde fuera, creando una instancia de la misma, pero claro, en cada archivo que necesitemos acceder a esa clase deberemos crear una nueva instancia y eso al final hace que la aplicación consuma más recursos de lo habitual.

Para evitar crear instancias de una clase en cada programa que requiera acceder a ella se usa el patron Singleton

Patron Singleton

- Constructor de ámbito privado o protegido
- Contenedor para guardar la instancia que creemos
- Método estático donde crearemos la instancia siempre que no este creada
- Redefinir el método mágico __clone para evitar que la instancia sea clonada

singleton.php

```
<?php
class patronSingleton
{  private  $Idb;       private  $filas = array();     private   static $instancia;  // contenedor de la instancia
  private function  __construct()  //  un constructor privado evita crear nuevos objetos desde fuera de la clase
    {  $this->Idb = new PDO( "mysql:host=localhost; dbname=database“ , 'user‘ , 'password‘);  }

  public  static  function   singleton()              //método singleton que crea instancia sí no está creada
    {  if (!isset(self::$instancia)) {
         $miclase = __CLASS__;
         self::$instancia = new $miclase;
        }
      return self::$instancia;
     }
  public  function   __clone()    // Evita que el objeto se pueda clonar
   {   trigger_error('La clonación de este objeto no está permitida',    E_USER_ERROR);    }

   public  function   usuarios()              //método con el que obtenemos datos de la tabla usuarios
    {  $consulta = $this->Idb->prepare("SELECT * FROM  usuarios");
      $consulta->execute();
      if ($consulta->rowCount()>0) :
        {   while($row=$consulta->fetch())    {    $this->filas[] = $row;    }
           return $this->filas;
        }
     }
}
?>
```

Veamos como usar la clase patronSingleton

```php
<?php
require_once   'singleton.php';   //incluimos el archivo que contiene nuestra clase

$conBD = patronSingleton::singleton();   //el metodo singleton es quien crea la instancia de la clase

$usuario = $conBD->usuarios();   //accedemos al método usuarios para obtener datos de una tabla

foreach ($usuario as $fila)      //  mostramos  los datos  de la tabla
{   ……
}
//  si  se   crean  instancias con new  dará error,  ya que el constructor es de ámbito privado
$conBD_2  = new patronSingleton() ;  //  Error

//  La creación de nuevas instancias  devuelven  la instancia ya creada  (  $conBD )
$conBD_2  = patronSingleton::singleton();

//  Si  se quiere clonar el objeto tampoco nos deja  y da   error
$conBD_2  = clone $conBD;
$otro_usuario = $conBD_2->usuarios();

?>
```