Universidad Nacional Abierta y a Distancia
Vicerrectoría Académica y de Investigación
Formato guía para instalación de herramientas de conexión PHP a base de datos.

CRUD: LA BASE DE LA GESTIÓN DE DATOS

El concepto CRUD está estrechamente vinculado a la gestión de datos digitales. CRUD hace referencia a **un acrónimo** en el que se reúnen las primeras letras de las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos:

- **C**reate (Crear registros)
- **R**ead bzw. **R**etrieve (Leer registros)
- **U**pdate (Actualizar registros)
- **D**elete bzw. **D**estroy (Borrar registros)

En pocas palabras, CRUD resume las funciones requeridas por un usuario para crear y gestionar datos. Varios procesos de gestión de datos están basados en CRUD, en los que **dichas operaciones están específicamente adaptadas a los requisitos del sistema y de usuario**, ya sea para la gestión de bases de datos o para el uso de aplicaciones. Para los expertos, las operaciones son las herramientas de acceso típicas e indispensables para comprobar, por ejemplo, los problemas de la base de datos, mientras que para los usuarios, CRUD significa crear una cuenta (créate) y utilizarla (read), actualizarla (update) o borrarla (delete) en cualquier momento. Dependiendo de la configuración regional, las operaciones CRUD pueden implementarse de diferentes maneras, como lo muestra la siguiente tabla:

CRUD	SQL	RESTful HTTP	XQuery
Create	INSERT	POST, PUT	insert
Read	SELECT	GET, HEAD	copy/modify/return
Update	UPDATE	PUT, PATCH	replace, rename
Delete	DELETE	DELETE	delete

Frameworks CRUD: capa de acceso a las bases de datos

Si los objetos individuales son **visualizados por medio de una interfaz gráfica** y modificados con las llamadas operaciones CRUD, entonces se habla de un framework CRUD o de un CRUD grid. Por lo general, se trata de **interfaces HTML**. Un framework CRUD demanda varios pasos de transacción, de forma que los datos no se recogen una vez se han introducido, sino que es necesario pulsar la opción “Guardar” o “Continuar”. Las operaciones de un framework CRUD pueden aplazarse para ser ejecutadas en diferentes plazos, sin que los datos de dichos periodos de tiempo se bloqueen para otros usuarios. Este hecho resulta de gran importancia para **sistemas multiusuario**, pues permite que varias personas lean los mismos datos al mismo tiempo.

Para llevar a cabo las operaciones se utilizan las denominadas capas de persistencia, que, por lo general, pueden ser implementadas o están contenidas en forma de extensiones (módulos) en el framework. Estas rompen con la representación relacional y tabular de la totalidad de los datos para, en su lugar, **presentarlos en un nivel orientado a objetos**. Los frameworks CRUD facilitan el acceso al sistema de bases de datos y son utilizados tanto en el desarrollo como en el uso de aplicaciones. Existen numerosos frameworks con un concepto CRUD basados en diferentes lenguajes y plataformas. A continuación, presentamos algunos ejemplos:

Lenguaje o plataforma	Framework
Java	JDBC (The Java Database Connectivity), Hibernate, JBoss Seam, Isis
PHP	Yii, CakePHP, Zikula, Symfony, TYPO3 Flow
Perl	Catalyst, Gantry
Python	Django, SQLAlchemy, web2py
Groovy	Grails
.NET	NHibernate, ADO.NET/Entity Framework
Ruby	Ruby on Rails
JavaScript	Backbone.js, AngularJS

Cómo desarrollar un CRUD PHP grid para tu base de datos

En la siguiente parte de este artículo te mostraremos cómo crear **una interfaz de arranque para MySQL**, el sistema de base de datos más utilizado, que permita el acceso a través operaciones CRUD. Adicionalmente podrás crear la operación “create”. Para manipular la base de datos se utiliza, en este caso, el lenguaje de script del lado del servidor **PHP** y la extensión **PHP Data Objects (PDO)**.

1. El primer paso consiste en crear una tabla de base de datos simple que pueda ser manipulada con operaciones CRUD a lo largo de este tutorial. Para ello, importa la siguiente tabla de ejemplo en tu base de datos MySQL:

```
CREATE TABLE `customers` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`name` VARCHAR( 100 ) NOT NULL ,
`email` VARCHAR( 100 ) NOT NULL ,
`mobile` VARCHAR( 100 ) NOT NULL
) ENGINE = INNODB;
```

La tabla se utiliza para recopilar información del usuario como nombre, correo electrónico y número de teléfono. A cada entrada se le asigna automáticamente una **clave principal** (AUTO_INCREMENT PRIMARY KEY), es decir un identificador único.

2. A continuación, es necesario regular las conexiones de apertura y de cierre de la base de datos. Crea un archivo PHP con el nombre database.php y añade la siguiente secuencia de comandos con la clase “Database” para gestionar las conexiones de la base de datos.

```
<?php
class Database
{
private static $dbName = 'nombre_de_la_basededatos';
private static $dbHost = 'localhost';
private static $dbUsername = 'nombre_de_usuario';
private static $dbUserPassword = 'contraseña';

private static $cont = null;

public function __construct() {
die('Init-Función no permitida');
}

public static function connect() {
// Permitir solo una conexión para la totalidad del acceso
if ( null == self::$cont )
{
  try
  {
    self::$cont = new PDO(
"mysql:host=".self::$dbHost.";"."dbname=".self::$dbName, self::$dbUsername,
self::$dbUserPassword);
  }
  catch(PDOException $e)
  {
    die($e->getMessage());
  }
}
return self::$cont;
}

public static function disconnect()
{
self::$cont = null;
}
}
```

Para que puedas utilizar la clase definida en el documento y lograr el acceso a la base de datos con PDO, necesitas los valores exactos de los elementos **$dbName** (nombre de la base de datos utilizada), **$dbHost** (nombre del host donde se ejecutará la base de datos, por lo general localhost como en el ejemplo), **$dbUsername** (nombre del usuario que accede) y

$dbUserPassword (contraseña del usuario que accede).

En esta secuencia de comandos se le asignan tres funciones a la clase “base de datos”: __construct(), **el constructor de la clase** que le recuerda a los usuarios que la inicialización (es decir, la asignación del valor inicial o de inicio) no está permitida. connect() hace referencia a la función principal de la clase que controla el establecimiento de la conexión y, por último, disconnect() que se encarga de finalizar la conexión.

3. Debido a que las operaciones CRUD solo pueden utilizarse con la correspondiente superficie de contacto, es necesario crear el grid base utilizando Twitter Bootstrap. La versión actual de este framework puede descargarse en su página web oficial. Descomprime Bootstrap en el mismo directorio donde se encuentra database.php y crea un nuevo archivo con el nombre index.php. Ahora puedes crear la interfaz:

```
<!DOCTYPE html>
<html lang="de">
<head>
<meta charset="utf-8">
<link href="css/bootstrap.min.css" rel="stylesheet">
</head>

<body>
<div class="container">
<div class="row">
<h3>My CRUD PHP grid</h3>
</div>
<div class="row">
<table class="table table-striped table-bordered">
<thead>
  <tr>
    <th>Nombre</th>
    <th>Correo electrónico </th>
    <th>Móvil</th>
  </tr>
</thead>
<tbody>
<?php
include 'database.php';
$pdo = Database::connect();
$sql = 'SELECT * FROM customers ORDER BY id DESC';
foreach ($pdo->query($sql) as $row) {
echo '<tr>';
echo '<td>' . $row['name'] . '</td>';
echo '<td>' . $row['email'] . '</td>';
echo '<td>' . $row['mobile'] . '</td>';
echo '</tr>';
}
```

```
Database::disconnect();
?>
</tbody>
</table>
</div>
</div> <!-- /container -->
</body>
</html>
```

En la sección <head>, se encuentran los archivos CSS y JavaScript de Bootstrap; en la sección <body> se encuentra el anteriormente creado database.php, incluyendo los intentos para establecer una conexión POD (Database::connect()) y sus datos relacionados (SELECT). Adicionalmente, el archivo contiene **la tabla HTML** <table> con las opciones nombre, correo electrónico y móvil (que también son almacenados en la base de datos).

El código genera **el formulario HTML** que permite personalizar la información en los campos nombre, correo electrónico y móvil. Para cada casilla se crea una variable PHP que, en combinación con el fragmento de código que presentamos a continuación (se debe añadir antes del código HTML en el archivo create.php), genera mensajes de error cuando no se realiza ninguna entrada en el campo correspondiente:

```
<?php

require 'database.php';

if ( !empty($_POST)) {
// Detectar errores de validación
$nameError = null;
$emailError = null;
$mobileError = null;

// Capturar valores de entrada
$name = $_POST['name'];
$email = $_POST['email'];
$mobile = $_POST['mobile'];

// Validar entrada
$valid = true;
if (empty($name)) {
$nameError = Por favor, introduce tu nombre';
$valid = false;
}

if (empty($email)) {
$emailError = 'Por favor, introduce una dirección de correo electrónico';
$valid = false;
```

```
} else if ( !filter_var($email,FILTER_VALIDATE_EMAIL) ) {
$emailError = 'Por favor, introduce una dirección de correo electrónico';
$valid = false;
}

if (empty($mobile)) {
$mobileError = 'Por favor, introduce tu número de móvil';
$valid = false;
}

// Daten eingeben
if ($valid) {
    $pdo = Database::connect();
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO customers (name,email,mobile) values(?, ?, ?)";
    $q = $pdo->prepare($sql);
    $q->execute(array($name,$email,$mobile));
    Database::disconnect();
    header("Location: index.php");
}
    }
?>
```

De esta forma habrás creado una página create.php a la que se accede haciendo clic en el botón crear y que permite al usuario introducir su información. El script se asegura de comprobar **que se hayan introducido todos los datos** y, en caso contrario, de **mostrar los respectivos errores de validación**; así, no solo se mostrarán mensajes **para avisar al usuario de que los datos serán enviados** a la base de datos, sino también **mensajes de error** cuando la entrada sea incorrecta.

Enlace para más información

https://www.1and1.es/digitalguide/paginas-web/desarrollo-web/crud-las-principales-operaciones-de-bases-de-datos/