**Taller práctico.**

**Actividad N 25.**

**MVC\_CRUD parte 2.**

**Objetivo de la actividad:** Codificar las interfaces gráficas y módulos usando PHP y el servidor apache aplicando el concepto de MVC\_CRUD

**Paso 6.**

Continuando con el ejercicio se construyen los atributos y métodos para el modelo, se crea el modelo, proveedor.php, este método selecciona todas las tuplas de la tabla proveedor en caso de error se muestra por pantalla de igual manera. Recuerda que los mismos deben ubicarse dentro de la carpeta model.

|  |
| --- |
| proveedor.php |
| <?php  class proveedor  {  //Atributo para conexión a SGBD  private $pdo;  //Atributos del objeto proveedor  public $nit;  public $razonS;  public $dir;  public $tel;  //Método de conexión a SGBD.  public function \_\_CONSTRUCT()  {  try  {  $this->pdo = Database::Conectar();  }  catch(Exception $e)  {  die($e->getMessage());  }  }  //Este método selecciona todas las tuplas de la tabla  //proveedor en caso de error se muestra por pantalla |

**Paso 7.**

Se construye el método listar, este método obtiene los datos del proveedor a partir del NIT, utilizando SQL.

|  |
| --- |
| proveedor.php (continuación código paso 6) |
| public function Listar()  {  try  {  $result = array();  //Sentencia SQL para selección de datos.  $stm = $this->pdo->prepare("SELECT \* FROM proveedor");  //Ejecución de la sentencia SQL.  $stm->execute();  //fetchAll — Devuelve un array que contiene todas las filas del conjunto  //de resultados  return $stm->fetchAll(PDO::FETCH\_OBJ);  }  catch(Exception $e)  {  //Obtener mensaje de error.  die($e->getMessage());  }  } |

**Paso 8.**

Este código complementa la acción solicitada anteriormente, ya que usa sentencia SQL para selección de datos utilizando la cláusula Where para especificar el nit del proveedor.

|  |
| --- |
| proveedor.php (continuación código paso 7) |
| //Este método obtiene los datos del proveedor a partir del nit  //utilizando SQL.  public function Obtener($nit)  {  try  {  //Sentencia SQL para selección de datos utilizando  //la cláusula Where para especificar el nit del proveedor.  $stm = $this->pdo->prepare("SELECT \* FROM proveedor WHERE nit = ?");  //Ejecución de la sentencia SQL utilizando el parámetro nit.  $stm->execute(array($nit));  return $stm->fetch(PDO::FETCH\_OBJ);  } catch (Exception $e)  {  die($e->getMessage());  }  } |

**Paso 9.**

Este código permite eliminar la tupa proveedor dado un nit. Utiliza una sentencia SQL para eliminar una tupla con la cláusula Where.

|  |
| --- |
| proveedor.php (continuación código paso 8) |
| //Este método elimina la tupla proveedor dado un nit.  public function Eliminar($nit)  {  try  {  //Sentencia SQL para eliminar una tupla utilizando  //la cláusula Where.  $stm = $this->pdo->prepare("DELETE FROM proveedor WHERE nit = ?");  $stm->execute(array($nit));  } catch (Exception $e)  {  die($e->getMessage());  }  } |

**Paso 10.**

Este código actualiza una tupla a partir de la cláusula Where y el nit del proveedor.

|  |
| --- |
| proveedor.php (continuación código paso 9) |
| //Método que actualiza una tupla a partir de la cláusula  //Where y el nit del proveedor.  public function Actualizar($data)  {  try  {  //Sentencia SQL para actualizar los datos.  $sql = "UPDATE proveedor SET  razonS = ?,  dir = ?,  tel = ?  WHERE nit = ?";  //Ejecución de la sentencia a partir de un arreglo.  $this->pdo->prepare($sql)  ->execute(  array(  $data->razonS,  $data->dir,  $data->tel,  $data->nit  )  );  } catch (Exception $e)  {  die($e->getMessage());  }  } |

**Paso 11.**

Este método registra un nuevo proveedor a la tabla.

|  |
| --- |
| proveedor.php (continuación código paso 10) |
| //Método que registra un nuevo proveedor a la tabla.  public function Registrar(proveedor $data)  {  try  {  //Sentencia SQL.  $sql = "INSERT INTO proveedor (nit,razonS,dir,tel)  VALUES (?, ?, ?, ?)";  $this->pdo->prepare($sql)  ->execute(  array(  $data->nit,  $data->razonS,  $data->dir,  $data->tel,  )  );  } catch (Exception $e)  {  die($e->getMessage());  }  }  } |

Como actividad debes:

1. Crear los elementos descritos desde el paso 6, hasta el paso 11. Has captura de pantalla según la siguiente tabla: (recuerda hacer la descripción de cada paso empleado una buena redacción)

|  |
| --- |
| Captura de pantalla paso6 |
| Creamos un archivo dentro de la carpeta model llamado “proveedor.php”, dentro de este creamos es una clase llamada proveedor. Esta clase tiene los siguientes componentes:  Atributos: La clase proveedor tiene los siguientes atributos:  $pdo: Este atributo es privado y se utiliza para la conexión a un sistema de gestión de bases de datos (SGBD).  $nit: Este atributo es público y representa el número de identificación tributaria del proveedor.  $razonS: Este atributo es público y representa la razón social del proveedor.  $dir: Este atributo es público y representa la dirección del proveedor.  $tel: Este atributo es público y representa el número de teléfono del proveedor.  Método de conexión a SGBD: La clase proveedor tiene un método llamado CONSTRUCT(), que se utiliza para establecer la conexión con un SGBD. En este caso, el método intenta establecer una conexión utilizando la clase Database::Conectar(). Si ocurre algún error durante la conexión, se captura la excepción y se muestra un mensaje de error. |
| Captura de pantalla paso7 |
| De pues creamos un método llamado Listar() que pertenece a la clase proveedor. Este método se utiliza para obtener todos los registros de la tabla proveedor en la base de datos.  El método Listar() utiliza una sentencia SQL para seleccionar todos los datos de la tabla proveedor. Luego, ejecuta la sentencia SQL y devuelve un array que contiene todas las filas del conjunto de resultados.  El array devuelto por el método Listar() contiene objetos que representan cada fila de la tabla proveedor. Cada objeto tiene propiedades que corresponden a las columnas de la tabla. |
| Captura de pantalla paso8 |
| Creamos otro método llamado Obtener($nit). Este método se utiliza para obtener los datos de un proveedor específico de la tabla proveedor de la base de datos.  El método Obtener($nit) utiliza una sentencia SQL para seleccionar los datos de la tabla proveedor que corresponden al valor del parámetro $nit. Luego, ejecuta la sentencia SQL y devuelve un objeto que representa la fila correspondiente a $nit.  El objeto devuelto por el método Obtener($nit) tiene propiedades que corresponden a las columnas de la tabla proveedor. |
| Captura de pantalla paso9 |
| Ahoara un método llamado Eliminar($nit). Este método se utiliza para eliminar una tupla de la tabla proveedor.  El método Eliminar($nit) utiliza una sentencia SQL para eliminar la tupla de la tabla proveedor que corresponde al valor del parámetro $nit. Luego, ejecuta la sentencia SQL y elimina la tupla correspondiente. |
| Captura de pantalla paso10 |
| Continuando ahora creamos un método llamado Actualizar($data). Este método se utiliza para actualizar los datos de un proveedor específico en la tabla proveedor.  El método Actualizar($data) utiliza una sentencia SQL para actualizar los datos de la tabla proveedor que corresponden al valor del parámetro $data. La sentencia SQL actualiza los valores de las columnas razonS, dir, y tel con los valores correspondientes en $data. La sentencia SQL también utiliza la cláusula WHERE para especificar el proveedor que se debe actualizar, utilizando el valor del atributo $nit.  Luego, el método ejecuta la sentencia SQL y actualiza los datos correspondientes en la tabla proveedor. |
| Captura de pantalla paso11 |
| Y por ultimo un método llamado Registrar(proveedor $data) Este método se utiliza para registrar un nuevo proveedor en la tabla proveedor de una base de datos.  El método Registrar(proveedor $data) utiliza una sentencia SQL para insertar los datos del proveedor en la tabla proveedor. La sentencia SQL utiliza la cláusula VALUES para especificar los valores que se deben insertar en las columnas nit, razonS, dir, y tel. Los valores se pasan como parámetros al método execute().  Luego, el método ejecuta la sentencia SQL y registra los datos del proveedor en la tabla proveedor. |

1. ¿para que se usa class proveedor? Explica en un párrafo de 57 palabras.

La clase "Proveedor" en el código se utiliza para interactuar con una tabla de proveedores en de la base de datos. Esta clase tiene métodos para realizar operaciones comunes en la tabla de proveedores, como listar todos los proveedores, obtener un proveedor por su NIT (Número de Identificación Tributaria), eliminar un proveedor por su NIT, actualizar los datos de un proveedor y registrar un nuevo proveedor en la tabla. Además, utiliza una conexión a la base de datos para realizar estas operaciones.

1. ¿para que se usa function \_\_CONSTRUCT()? Explica en un párrafo de 57 palabras.

La función \_\_CONSTRUCT() es un método especial que se utiliza en PHP para definir el constructor de una clase. El constructor se llama automáticamente cuando se crea un objeto de la clase y se utiliza para inicializar los atributos del objeto. El método \_\_CONSTRUCT() se utiliza para establecer la conexión con una base de datos, inicializar los atributos de la clase y realizar cualquier otra tarea necesaria antes de que el objeto pueda ser utilizado. El método \_\_CONSTRUCT() también puede aceptar argumentos que se utilizan para inicializar los atributos del objeto.

1. ¿Qué es el cakePHP? Explica en un párrafo de 500 palabras según sea el caso.

CakePHP es un marco de desarrollo rápido y de código abierto para aplicaciones web construido en PHP. Sigue el enfoque modelo-vista-controlador (MVC) y está escrito en PHP, modelado según los conceptos de Ruby on Rails. CakePHP se distribuye bajo la licencia MIT. El objetivo principal de CakePHP es simplificar y acelerar el desarrollo de aplicaciones web, reduciendo la cantidad de código necesario. Proporciona una amplia gama de características y herramientas integradas, como validación de entrada, protección contra CSRF, prevención de inyección SQL y más. Además, CakePHP tiene una comunidad activa y amigable que contribuye al proyecto y ofrece soporte.

CakePHP se basa en el patrón de diseño MVC, que separa la lógica de la aplicación en tres componentes principales: el modelo, la vista y el controlador. El modelo se encarga del acceso a los datos y la lógica empresarial. La vista se encarga de la presentación de los datos al usuario. El controlador maneja las solicitudes del usuario y coordina las interacciones entre el modelo y la vista.

Una de las ventajas clave de CakePHP es su capacidad para generar código automáticamente a través de la generación de código y las características de andamiaje (scaffolding). Esto permite a los desarrolladores crear rápidamente prototipos y comenzar a desarrollar sin tener que escribir mucho código inicialmente. Además, CakePHP sigue convenciones claras y predecibles, lo que facilita el desarrollo y mejora la legibilidad del código.

CakePHP también ofrece una amplia gama de componentes y ayudantes integrados que facilitan tareas comunes, como el manejo de formularios, AJAX, autenticación, seguridad, caché y más. Estos componentes ayudan a los desarrolladores a escribir menos código repetitivo y acelerar el proceso de desarrollo.