**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 |
| Proveedor.php |
| Captura de pantalla paso7 |
| Proveedor.php |
| Captura de pantalla paso8 |
| Proveedor.php |
| Captura de pantalla paso9 |
| Proveedor.php |
| Captura de pantalla paso10 |
| Proveedor.php |
| Captura de pantalla paso11 |
| Proveedor.php |

1. ¿para que se usa class proveedor? Explica en un párrafo de 57 palabras.
   1. Se usa para almacenar todos los métodos o acciones las cuales va a usar el proveedor.
2. ¿para que se usa function \_\_CONSTRUCT()? Explica en un párrafo de 57 palabras.
   1. Es un termino reservado de PHP que permite el \_\_construct() permite crear un método automáticamente de cuando se crea un objeto. Además de que permite inyectar parámetros para construir el objeto. En este caso lo usamos para construir la conexión a la base de datos.
3. ¿Qué es el cakePHP? Explica en un párrafo de 500 palabras según sea el caso. Emplea un gráfico para entender el mismo.
   1. CakePHP es un framework de desarrollo web de código abierto basado en el patrón Modelo-Vista-Controlador (MVC) para PHP. Proporciona una estructura y una serie de herramientas y bibliotecas predefinidas para acelerar el proceso de desarrollo web en PHP, al tiempo que sigue las mejores prácticas y estándares de la industria. Con CakePHP, los desarrolladores pueden construir aplicaciones web de manera eficiente y escalable, al mismo tiempo que mantienen un código limpio y fácil de mantener. El objetivo de CakePHP es permitir a los desarrolladores crear aplicaciones web de alta calidad en PHP de manera rápida y fácil.