República Bolivariana de Venezuela.

Ministerio del Poder Popular Para La Educación.

Universidad Nororiental Privada Gran Mariscal de Ayacucho.

Facultad de Ingeniería de sistemas

Ingeniería de software II.

6to Semestre.

**Diccionario de datos**

Profesora: Integrantes:

Thays Parra Emilmar Cuarez 28.619.685

Lorena Moujalli 28.255.372

Jairo Narvaez 28.683.334

Paola Salazar 30.791.332

Luz Sanchez 27.032.910

Puerto Ordaz, 2022

|  |  |
| --- | --- |
| Función | Definición |
| public function \_\_construct(){              $this->model=new Modelo();          } | Función constructora de la clase ModeloController |
| static function iniciar(){            require\_once("vista/login.php");          } | Función estática de la clase ModeloController que se encarga de redireccionar a la página de login.php |
| static function inicio(){            require\_once("vista/index.php");          } | Función estática de la clase ModeloController que se encarga de redireccionar a la página index.php que es la página de inicio luego de iniciar sesión |
| static function restablecer(){            require\_once("vista/restablecer.php");       } | Función estática de la clase ModeloController que se encarga de redireccionar a la página restablecer.php que es la página de restablecimiento de contraseña |
| public static function index(){              $datos=new Modelo();              $data=$datos->traerIdSesiones("sesiones");              $datix=$datos->traerIdUser("sesiones",$data);              $dato=$datos->mostrar("tarea",0,$datix);              require\_once("vista/tareas.php");             } | Función estática de la clase ModeloController que se encarga de redireccionar a la página tareas.php y se encarga de cargar todas las tareas de un usuario en específico. |
| static function nuevo(){             require\_once("vista/nuevo.php");          } | Función estática de la clase ModeloController que se encarga de redireccionar a la página de nuevo.php que es la página de registro de usuario |
| static function guardar(){            $titulo=$\_REQUEST['titulo'];            $mensaje=$\_REQUEST['mensaje'];            $color=$\_REQUEST['color'];            $fecha\_inicio=date('d/m/y');            $fecha\_finalizacion=$\_REQUEST['fecha'];              $condicion=0;              $usuario=new Modelo();              $datu=$usuario->traerIdSesiones("sesiones");              $datix=$usuario->traerIdUser("sesiones",$datu);              $data="'".$titulo."','".$mensaje."','".$fecha\_inicio."','".$fecha\_finalizacion."','".$color."',".$condicion.",". $datix;                  $dato=$usuario->insertar("tarea",$data);                header('location:'.'index.php?n=index');           } | Función estática de la clase ModeloController que se encarga de recibir todos los datos que se ingresaron para registrar una tarea y se envían a la clase Modelo para registrarlo en la base de datos y se redirecciona a la función llamada index para que se carguen todas las tareas registradas hasta el momento |
| static function guardaru(){              $nombre=$\_REQUEST['nombre'];              $apellido=$\_REQUEST['apellido'];              $n\_usuario=$\_REQUEST['nombredeusuario'];              $contrasenia=$\_REQUEST['contrasenia'];              $correo=$\_REQUEST['correo'];              $data="'".$nombre."','".$apellido."','".$n\_usuario."','".$contrasenia."','".$correo."'";              $usuario=new Modelo();              if($usuario->validar\_User\_existente("usuario","n\_usuario='".$n\_usuario."'","correo='".$correo."'")){                header('location:'.ModeloController::nuevo());                 echo "<script>alert('el nombre de usuario: $n\_usuario o el correo: $correo ya estan siendo utilizados');</script>";              }else{                $dato=$usuario->insertar("usuario",$data);                 header('location:'.'index.php?n=iniciar');              }           } | Función estática de la clase ModeloController que se encarga de recibir todos los datos que se ingresaron para registrar un usuario y se envían a la clase Modelo para registrarlo en la base de datos y se redirecciona a la función llamada iniciar para cargar la pagina de inicio de sesión |
| static function buscar(){            $titulo=$\_REQUEST['titulo'];            $data="titulo='".$titulo."'";            $usuario=new Modelo();              $dato=$usuario->mostrar2("tarea",$data);              require\_once("vista/tareas.php");         } | Función estática de la clase ModeloController que recibe el título de la tarea que se va a buscar y luego muestra esa tarea encontrada. Después redirecciona a la página tarea.php |
| static function actualizaru(){              $correo=$\_REQUEST['correo'];            $contrasenia=$\_REQUEST['contrasenia'];            $data   = "contrasenia='".$contrasenia."'";             $datos=new Modelo();             if($datos->validar\_User\_existente2("usuario","correo= '".$correo."'" )){                $mostrar=$datos->traerid("usuario","correo= '".$correo."'");                $dato=$datos->actualizar("usuario",$data,"id=".$mostrar);                header('location:'.ModeloController::index());                echo "<script>alert('La contraseña fue cambiada exitosamente');</script>";             }             else{              header('location:'.ModeloController::restablecer());              echo "<script>alert('El correo: $correo  no existe');</script>";             }              } | Función estática de la clase ModeloController que recibe los nuevos datos que se editaran del usuario, se llama a la función validar\_User\_existente para verificar si existe este usuario, si existe utilizamos la función traerid para traer el id del usuario a modificar. Y se manda los datos a la función actualizar de la clase Modelo y luego invoca la función index y se envía un mensaje diciendo que la contraseña fue actualiza exitosamente. En caso contrario que no exista el usuario se envía un mensaje diciendo que este no existe y se llama a la función restablecer |
| public static function login(){          $usuario=$\_REQUEST['n\_usuario'];          $contrasenia=$\_REQUEST['contrasenia'];           $datos=new Modelo();           $data="n\_usuario='".$usuario."' AND clave='".$contrasenia."'";           $dato=$datos->login("usuario",$data);           $datu=$datos->traerId("usuario",$data);           $datix=$datos->insertar("sesiones",$datu);          if($dato==true){            header('location:'.'index.php?n=inicio');          }          else{            echo "<script>alert('la contraseña/nombre de usuario no coinciden');</script>";            header('location:'.'index.php?n=iniciar');          }        } | Función estática de la clase ModeloController que se encarga de recibir todos los datos que se ingresaron para iniciar sesión y se envían a la clase Modelo para verificar que existen en la base de datos y se redirecciona a la función llamada inicio. Y en caso contrario que el usuario no esté registrado en la base de datos se envía un mensaje diciendo que la contraseña/nombre de usuario no coinciden |
| static function editar(){            $id=$\_REQUEST['id'];            $datos=new Modelo();            $dato=$datos->mostrar2("tarea","id=".$id);            require\_once("vista/editar.php");           } | función estática de la clase ModeloController que recibe el id de la tarea que se desea editar y muestra sus datos en la página editar.php |
| static function editar2(){            $id=$\_REQUEST['id'];            $datos=new Modelo();            $dato=$datos->mostrar2("tarea","id=".$id);            require\_once("vista/editar2.php");           } | función estática de la clase ModeloController que recibe el id de la tarea que se desea editar desde la página de archivados y muestra sus datos en la página editar2.php |
| static function actualizar(){              $id=$\_REQUEST['id'];               $titulo=$\_REQUEST['titulo'];              $mensaje=$\_REQUEST['mensaje'];              $color=$\_REQUEST['color'];              $fecha\_finalizacion=$\_REQUEST['fecha'];              $data="titulo='".$titulo."',descripcion='".$mensaje."',color='".$color."',fecha\_finalizacion='".$fecha\_finalizacion."'";               $datos=new Modelo();               $dato=$datos->actualizar("tarea",$data,"id=".$id);               header('location:'.'index.php?n=index');            } | función estática de la clase ModeloController que recibe los nuevos datos de la tarea que se editó, manda los datos a la función actualizar de la clase Modelo y luego invoca la función index. |
| static function actualizar2(){              $id=$\_REQUEST['id'];               $titulo=$\_REQUEST['titulo'];              $mensaje=$\_REQUEST['mensaje'];              $color=$\_REQUEST['color'];              $fecha\_finalizacion=$\_REQUEST['fecha'];              $data="titulo='".$titulo."',descripcion='".$mensaje."',color='".$color."',fecha\_finalizacion='".$fecha\_finalizacion."'";               $datos=new Modelo();               $dato=$datos->actualizar("tarea",$data,"id=".$id);               header('location:'.'index.php?n=paginaArch');            } | función estática de la clase ModeloController que recibe los nuevos datos de la tarea que se editó, manda los datos a la función actualizar de la clase Modelo y luego invoca la función paginaArch para que se muestre la página archivados con todas las tareas archivadas ahí. |
| static function archivar(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $dato=$datos->archivar("tarea","id=".$id,1);              header('location:'.'index.php?n=index');            } | función estática de la clase ModeloController que recibe el id de la tarea que se archivara y luego redirecciona a la función index. |
| static function sacarArch(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $datos2=new Modelo();//anotaciones               $data=$datos->traerIdSesiones("sesiones");              $datix=$datos->traerIdUser("sesiones",$data);              $dato=$datos->archivar("tarea","id=".$id,0);               $dato=$datos->mostrar("tarea","1",$datix);               $data=$datos2->traerIdSesiones("sesiones");//anotaciones               $datix=$datos2->traerIdUser("sesiones",$data);//anotaciones               $dato2=$datos2->archivar("anotaciones","id=".$id,0);//anotaciones                $dato2=$datos2->mostrar\_nota("anotaciones","1",$datix);//anotaciones              require\_once("vista/archivados.php");            } | función estática de la clase ModeloController que recibe el id de la tarea y la anotación que se sacaran de la página archivados y luego llama a la página archivados.php |
| static function paginaArch(){              $datos=new Modelo();              $datos2=new Modelo(); //anotaciones              $data=$datos->traerIdSesiones("sesiones");              $datix=$datos->traerIdUser("sesiones",$data);              $dato=$datos->mostrar("tarea",1,$datix);              $data=$datos2->traerIdSesiones("sesiones");//anotaciones              $datix=$datos2->traerIdUser("sesiones",$data);//anotaciones              $dato2=$datos2->mostrar\_nota("anotaciones",1,$datix);//anotaciones              require\_once("vista/archivados.php");            } | función estática de la clase ModeloController que carga todas las tareas y anotaciones que están archivadas y luego llama a la página archivados.php |
| static function papeleraArch(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $datos2=new Modelo(); //anotaciones              $dato=$datos->archivar("tarea","id=".$id,2);              $dato2=$datos2->archivar("anotaciones","id=".$id,2);              header('location:'.'index.php?n=paginaArch');            } | función estática de la clase ModeloController que manda la tarea y la anotación seleccionada a papelera. Por lo tanto, manda los datos a la función archivar a la cual le manda un 2 al final de su parámetro, lo que significa que se actualizara la tarea y anotación a condición=2, lo cual es que esta en papelera, y luego llama a la función paginaArch para que vuelva a mostrar las tareas y anotaciones que están archivadas. |
| static function papelera(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $dato=$datos->archivar("tarea","id=".$id,2);              header('location:'.'index.php?n=index');            } | función estática de la clase ModeloController que manda la tarea seleccionada a papelera. Por lo tanto, manda los datos a la función archivar a la cual le manda un 2 al final de su parámetro, lo que significa que se actualizara la tarea a condición=2, lo cual es que esta en papelera, y luego llama a la función index para que vuelva a mostrar las tareas que están en la página de tareas (sin ser archivadas y sin contar las que están en papelera). |
| static function sacarPapelera(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $datos2=new Modelo(); //anotaciones               $data=$datos->traerIdSesiones("sesiones");              $datix=$datos->traerIdUser("sesiones",$data);              $dato=$datos->archivar("tarea","id=".$id,0);               $dato=$datos->mostrar("tarea","2",$datix);               $data=$datos2->traerIdSesiones("sesiones"); //anotaciones               $datix=$datos2->traerIdUser("sesiones",$data); //anotaciones               $dato2=$datos2->archivar("anotaciones","id=".$id,0); //anotaciones                $dato2=$datos2->mostrar\_nota("anotaciones","2",$datix); //anotaciones              require\_once("vista/papelera.php");            } | función estática de la clase ModeloController que recibe el id de la tarea y la anotación que se sacaran de la página papelera y luego llama a la página papelera.php |
| static function paginaPapelera(){              $datos=new Modelo();              $datos2=new Modelo(); // anotaciones              $data=$datos->traerIdSesiones("sesiones");              $datix=$datos->traerIdUser("sesiones",$data);              $dato=$datos->mostrar("tarea",2,$datix);              $data=$datos2->traerIdSesiones("sesiones"); //anotaciones              $datix=$datos2->traerIdUser("sesiones",$data); //anotaciones              $dato2=$datos2->mostrar\_nota("anotaciones",2,$datix); //anotaciones              require\_once("vista/papelera.php");            } | función estática de la clase ModeloController que carga todas las tareas y anotaciones que están en papelera y luego llama a la página papelera.php |
| static function eliminar(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $dato=$datos->eliminar("tarea","id=".$id);              header('location:'.'index.php?n=index');           } | función estática de la clase ModeloController que elimina la tarea que se está seleccionando y luego se redirecciona a la función index |
| static function eliminarArchivado(){              $id=$\_REQUEST['id'];              $datos=new Modelo();              $dato=$datos->eliminar("tarea","id=".$id);              header('location:'.'index.php?n=paginaArch');           } | función estática de la clase ModeloController que elimina la tarea que se está seleccionando y luego se redirecciona a la función paginaArch |
| public function insertar($tabla, $data){          $consulta="insert into ".$tabla." values(null,".$data.")";          $resultado=$this->db->query($consulta);          if($resultado){              return true;          }else{              return false;          }      } | Función de la clase Modelo que se encarga de recibir el nombre de la tabla y el valor de los datos que se van a ingresar a la base de datos. Retorna true si la consulta se realizó con éxito y false en caso contrario. |
| public function traerId($tabla, $data){          $consulta="select id from ".$tabla." where ".$data.";";          $resul=$this->db->query($consulta);          $row = $resul->fetch();          $res = $row[0];          return $res;      } | función de la clase Modelo que se encarga de recibir el nombre de la tabla y la condición que se tiene que tener en cuenta al hacer la consulta. Retorna el id que corresponda a los datos recibidos por parámetros. Manda esta consulta a la base de datos y retorna un número. |
| public function traerIdSesiones($tabla){          $consulta="select MAX(id) from ".$tabla.";";          $resul=$this->db->query($consulta);          $row = $resul->fetch();          $res = $row[0];          return $res;      } | función de la clase Modelo que recibe el nombre de la tabla de donde se quiere obtener el ultimo id registrado. Retorna ese id encontrado en la base de datos. |
| public function traerIdUser($tabla, $data){          $consulta="select id\_usuario from ".$tabla." where id=".$data.";";          $resul=$this->db->query($consulta);          $row = $resul->fetch();          $res = $row[0];          return $res;      } | función de la clase Modelo que se encarga de recibir el nombre de la tabla y la condición que se tiene que tener en cuenta al hacer la consulta. Retorna el id\_usuario que corresponda a los datos recibidos por parámetros. Manda esta consulta a la base de datos y retorna el id\_usuario encontrado. |
| public function mostrar($tabla,$condicion,$id){          $consul="select \* from ".$tabla." where condicion=".$condicion." and id\_usuario=".$id;          $resul=$this->db->query($consul);          while($filas=$resul->FETCHALL(PDO::FETCH\_ASSOC)){              $this->datos[]=$filas;          }          return $this->datos;      } | función de la clase Modelo que recibe la tabla, la condición y el id del usuario que se tiene que tomar en cuenta al realizar la consulta. Consiste en retornar todos los datos encontrados que sean del usuario recibido por parámetro y cumpla con la condición. |
| public function mostrar2($tabla,$condicion){          $consul="select \* from ".$tabla." where ".$condicion;          $resul=$this->db->query($consul);          while($filas=$resul->FETCHALL(PDO::FETCH\_ASSOC)){              $this->datos[]=$filas;          }          return $this->datos;      } | función de la clase Modelo que recibe la tabla y la condición que se tiene que tomar en cuenta al realizar la consulta. Consiste en retornar todos los datos encontrados que sean de la tabla recibida por parámetro y cumpla con la condición. |
| public function actualizar($tabla, $data, $condicion){          $consulta="update ".$tabla." set ". $data ." where ".$condicion;          $resultado=$this->db->query($consulta);          if($resultado){              return true;          }else{              return false;          }      } | función de la clase Modelo que recibe la tabla, los datos a actualizar y la condición que se tiene que tomar en cuenta al realizar la consulta. Manda los datos a la base de datos y retornar true si se realizo la consulta y false en caso contrario. |
| public function archivar($tabla, $data,$condicion){          $consulta="update ".$tabla." set condicion=".$condicion." where ".$data;          $resultado=$this->db->query($consulta);          if($resultado){              return true;          }else{              return false;          }       } | función de la clase Modelo que recibe la tabla, los datos a actualizar y la condición que se tiene que tomar en cuenta al realizar la consulta. Consiste en cambiar la condición de la tarea o anotación. En caso de que la condición sea 1, quiere decir que la tarea esta en archivados, si es 2, es que ahora estará en papelera y 0 es que esta en la parte de tareas en general. Manda los datos a la base de datos y retornar true si se realizó la consulta y false en caso contrario. |
| public function eliminar($tabla, $condicion){          $eli="delete from ".$tabla." where ".$condicion;          $res=$this->db->query($eli);          if($res){              return true;          }else{              return false;          }      } | función de la clase Modelo que se encarga de eliminar un dato que este en la tabla y cumpla con la condición que se recibe por parámetro. |
| public function login($tabla, $condicion){          $consul="select \* from ".$tabla." where ".$condicion.";";          $resul=$this->db->query($consul);          if($resul){              return $resul->fetchAll(PDO::FETCH\_ASSOC);          }else{              return false;          }      } | Función de la clase Modelo que se encarga de recibir el nombre de la tabla y la condición que se tiene que tener en cuenta al hacer la consulta Consiste en verificar si los datos se encuentran en la tabla recibida por parámetro y cumpla con la condición y retornar esos datos. Y en caso contrario retornar false. |
| public function validar\_User\_existente($tabla, $condicion,$condicion2){          $consul="select \* from ".$tabla." where ".$condicion." OR ".$condicion2.";";          $resul=$this->db->query($consul);          if($resul){              return $resul->fetchAll(PDO::FETCH\_ASSOC);          }else{              return false;          }      } | Función de la clase Modelo que se encarga de recibir el nombre de la tabla y las condiciónes que se tiene que tener en cuenta al hacer la consulta Consiste en verificar si los datos se encuentran en la tabla recibida por parámetro y cumpla con una de las condiciones y retornar esos datos. Y en caso contrario retornar false. |
| public function validar\_User\_existente2($tabla, $condicion){          $consul="select \* from ".$tabla." where ".$condicion.";";          $resul=$this->db->query($consul);          if($resul){              return $resul->fetchAll(PDO::FETCH\_ASSOC);          }else{              return false;          }      } | Función de la clase Modelo que se encarga de recibir el nombre de la tabla y la condición que se tiene que tener en cuenta al hacer la consulta Consiste en verificar si los datos se encuentran en la tabla recibida por parámetro y cumpla con la condición y retornar esos datos. Y en caso contrario retornar false. |
| static function guardar\_nota(){      $descripcion=$\_REQUEST['descripcion\_anotaciones'];        $condicion=0;        $usuario=new Modelo();        $datu=$usuario->traerIdSesiones("sesiones");        $datix=$usuario->traerIdUser("sesiones",$datu);        $data="'".$descripcion."',".$condicion.",".$datix;          $dato2=$usuario->insertar("anotaciones",$data);          header('location:'.'index.php?n=index');     } | Función estática de la clase ModeloController que se encarga de recibir todos los datos que se ingresaron para registrar una anotación y se envían a la clase Modelo para registrarlo en la base de datos y se redirecciona a la función llamada index para que se carguen todas las anotaciones registradas hasta el momento |
| static function editar\_nota(){        $id=$\_REQUEST['id'];        $datos2=new Modelo();        $dato2=$datos2->mostrar\_nota2("anotaciones","id=".$id);        require\_once("vista/editar\_nota.php");       } | función estática de la clase ModeloController que recibe el id de la anotación que se desea editar y muestra sus datos en la página editar\_nota.php |
| static function editar\_nota2(){            $id=$\_REQUEST['id'];            $datos2=new Modelo();            $dato2=$datos2->mostrar\_nota2("anotaciones","id=".$id);            require\_once("vista/editar\_nota2.php");           } | función estática de la clase ModeloController que recibe el id de la anotación que se desea editar desde archivados y muestra sus datos en la página editar\_nota2.php |
| static function actualizar\_nota(){        $id=$\_REQUEST['id'];        $descripcion=$\_REQUEST['descripcion\_anotaciones'];        $data="descripcion\_anotaciones='".$descripcion."'";         $datos2=new Modelo();         $dato2=$datos2->actualizar("anotaciones",$data,"id=".$id);         header('location:'.'index.php?n=index');      } | función estática de la clase ModeloController que recibe los nuevos datos de la anotacion que se editó, manda los datos a la función actualizar de la clase Modelo y luego invoca la función index. |
| static function actualizar\_nota2(){        $id=$\_REQUEST['id'];        $descripcion=$\_REQUEST['descripcion\_anotaciones'];        $data="descripcion\_anotaciones='".$descripcion."'";         $datos2=new Modelo();         $dato2=$datos2->actualizar("anotaciones",$data,"id=".$id);         header('location:'.'index.php?n=index');      } | función estática de la clase ModeloController que recibe los nuevos datos de la anotacion que se editó, manda los datos a la función actualizar de la clase Modelo y luego invoca la función paginaArch para que se muestre la página archivados con todas las anotaciones archivadas ahí. |
| static function archivar\_nota(){      $id=$\_REQUEST['id'];      $datos2=new Modelo();      $dato2=$datos2->archivar("anotaciones","id=".$id,1);      header('location:'.'index.php?n=index');    } | función estática de la clase ModeloController que recibe el id de la anotacion que se archivara y luego redirecciona a la función index. |
| static function papelera\_nota(){      $id=$\_REQUEST['id'];      $datos2=new Modelo();      $dato2=$datos2->archivar("anotaciones","id=".$id,2);      header('location:'.'index.php?n=index');    } | función estática de la clase ModeloController que manda la anotacion seleccionada a papelera. Por lo tanto, manda los datos a la función archivar a la cual le manda un 2 al final de su parámetro, lo que significa que se actualizara la anotacion a condición=2, lo cual es que esta en papelera, y luego llama a la función index para que vuelva a mostrar las anotaciones que están en la página de tareas (sin ser archivadas y sin contar las que están en papelera). |
| static function eliminar\_nota(){        $id=$\_REQUEST['id'];        $datos2=new Modelo();        $dato2=$datos2->eliminar("anotaciones","id=".$id);        header('location:'.'index.php?n=index');     } | función estática de la clase ModeloController que elimina la anotacion que se está seleccionando y luego se redirecciona a la función index |
| static function eliminarArchivado\_nota(){        $id=$\_REQUEST['id'];        $datos2=new Modelo();        $dato2=$datos2->eliminar("anotaciones","id=".$id);        header('location:'.'index.php?n=paginaArch');     } | función estática de la clase ModeloController que elimina la anotacion que se está seleccionando y luego se redirecciona a la función paginaArch |
| public function mostrar\_nota($tabla,$condicion,$id){    $consul="select \* from ".$tabla." where condicion=".$condicion." and id\_usuario=".$id; $resul=$this->db->query($consul);   while($filas=$resul->FETCHALL(PDO::FETCH\_ASSOC)){        $this->datos2[]=$filas;       }        return $this->datos2;        } | función de la clase Modelo que recibe la tabla, la condición y el id del usuario que se tiene que tomar en cuenta al realizar la consulta. Consiste en retornar todos los datos encontrados que sean del usuario recibido por parámetro y cumpla con la condición. |
| public function mostrar\_nota2($tabla,$condicion){          $consul="select \* from ".$tabla." where ".$condicion;          $resul=$this->db->query($consul);          while($filas=$resul->FETCHALL(PDO::FETCH\_ASSOC)){              $this->datos2[]=$filas;          }          return $this->datos2;      } | función de la clase Modelo que recibe la tabla y la condición que se tiene que tomar en cuenta al realizar la consulta. Consiste en retornar todos los datos encontrados que sean de la tabla recibida por parámetro y cumpla con la condición. |
| static function actualizaru2(){              $nombre=$\_REQUEST['nombre'];            $apellido=$\_REQUEST['apellido'];            $n\_usuario=$\_REQUEST['n\_usuario'];            $correo=$\_REQUEST['correo'];            $contrasenia=$\_REQUEST['contrasenia'];            $data="nombre='".$nombre."',apellido='".$apellido."',n\_usuario='".$n\_usuario."',correo='".$correo. "',clave='".$contrasenia."'";             $datos=new Modelo();             $datu=$datos->traerIdSesiones("sesiones");        $datix=$datos->traerIdUser("sesiones",$datu);              $dato=$datos->actualizar("usuario",$data,"id=".$datix);              header('location:'.'index.php?n=personales');              } | Función estática de la clase ModeloController que actualiza los datos del usuario ya registrados anteriormente. |

Diccionario de datos de las variables:

|  |  |
| --- | --- |
| Variable | Definición |
| $model | Atributo De la clase ModeloController que cosiste en ser una instancia de la clase Modelo |
| $datos | Consiste en ser una instancia de la clase Modelo para poder insertar datos a la base de datos. |
| $data | Almacena los datos de la consulta que se desea mandar a la base de datos y en otras funciones sirve para contener el id de la sesión que se está ejecutando. |
| $datix | Devuelve el id del usuario que está en la sesión actual |
| $dato | Contiene un arreglo con todas las tareas que se deben mostrar. |
| $titulo | Almacena el título de la tarea que ingreso el usuario |
| $mensaje | Almacena el mensaje de la tarea que ingreso el usuario |
| $color | Almacena el color de la tarea que ingreso el usuario |
| $fecha\_incio | Almacena la fecha de inicio en que se está registrando la tarea. |
| $fecha\_finalizacion | Almacena la fecha de finalización de la tarea que ingreso el usuario |
| $condicion | Almacena la condición en la que se encuentra la tarea al momento. También almacena los datos de registro de usuario que se enviaran a la base de datos |
| $usuario | Consiste en ser una instancia de la clase Modelo para poder insertar datos. Tambien se usa en la función login para almacenar el nombre de usuario |
| $nombre | Contiene el nombre del usuario ingresado por el mismo usuario al momento del registro |
| $apellido | Contiene el apellido del usuario ingresado por el mismo usuario al momento del registro |
| $correo | Contiene el correo del usuario ingresado |
| $n\_usuario | Contiene el nombre de usuario ingresado |
| $contrasenia | Contiene la contraseña ingresada por el usuario |
| $mostrar | Contiene el id del usuario registrado |
| $datu | contiene el id de la sesión que se está ejecutando. |
| $id | contiene el id de la tarea seleccionada |
| $Modelo | Atributo de la clase Modelo que contiene un array |
| $db | Es un objeto PDO que contiene la información para acceder a la base de datos |
| $datos | Almacena el arreglo con toda la información que se trajo de la consulta a la base de datos para mostrar. |
| $consulta | Contiene la consulta SQL que se mandara a la base de datos |
| $resul | Almacena el resultado de la consulta a la base de datos |
| $row | Contiene la fila que retorna la consulta a la base de datos |
| $res | Contiene el resultado que almacena la fila en la posición 0. |
| $consul | Contiene la consulta SQL que se mandara a la base de datos |
| $res | Contiene la consulta SQL que se mandara a la base de datos para eliminar un elemento |
| let btn | Almacena la acción que se le ejerce al botón para abrir y cerrar el menú. |
| let modal | Almacena la acción que se le ejerce al botón para abrir y cerrar el modal de tareas. |
| let flex | Almacena la acción que cambiará cuando abran o cierren el modal de tareas |
| Let abrir | Se encarga de que al darle clic al botón de crear tareas se abra el modal. |
| Let cerrar | Se encarga de que al darle clic al botón de cerrar del modal, este se oculte. |
| $descripcion | Almacena la descripción de la anotación que ingreso el usuario |
| $dato2 | Contiene un arreglo con todas las anotaciones que se deben mostrar. (anotaciones) |
| $datos2 | Almacena el arreglo con toda la información que se trajo de la consulta a la base de datos para mostrar. (anotaciones) |
| $id | contiene el id de la anotación seleccionada |