Universidad Morelos Politécnica

EP3. SITIO WEB APLICANDO MVC

Programación Web

Prof. César Iván Ponciano Velasquez

Gonzalez Martinez Juan Antonio GMJO231476

Morales Soto Maria de los Angeles MSMO230605

ITI 7° D

07/11/2025

Índice

Introducción

En este documento se presenta la gestión de alumnos en MVC (Modelo, Vista y Controlador), mostrando el código y explicando que hace cada parte para poder registrar, consultar, editar y eliminar así como la conexión a la base de datos además de las pruebas para verificar que la gestión funcione correctamente.

Modelo

Para nuestro modelo **Figura 1** primero creamos una clase llamada “UserModel”, la variable $connection nos permitirá hacer la conexión a la base de datos que estamos usando en este caso “repositorios”, se crea un constructor para recibir la información posteriormente un método para poder insertar en nuestra tabla de la base de datos para esto usamos bind_param() que asocia las variables PHP a los ? del SQL y execute() que ejecuta la consulta y devuelva TRUE o FALSE si la inserción fue exitosa. El siguiente método es consultar, qué hace un select ay obtiene los registros de la tabla y devuelve el resultado para que se muestran los datos

```php
<?php

    //Crear una clase dek modelo
    class UserModel{
        private $connection;

        //Crear constructor para recibir la conexion
        public function __construct($connection){
            $this -> connection = $connection; //Se puede utizar cualquier palabra
        }

        // Metodo para insertar en la base de datos
        public function insertarUsuario($nombre, $apellidoP, $apellidoM, $usuario, $correo, $pass, $genero){

            $sql_statement = "INSERT INTO alumno (nombre, apellidoP, apellidoM, usuario,
            correo, pass, genero) VALUES (?,?,?,?,?,?,?)";

            // Preparar el statement
            $statement = $this -> connection -> prepare($sql_statement);
            $statement -> bind_param("sssssss",$nombre, $apellidoP, $apellidoM, $usuario, $correo, $pass, $genero);

            return $statement -> execute();

        }

        //Metodo para consultar los usuarios
        public function consultarUsuarios(){

            $sql_statement = "SELECT * FROM alumno";

            //Guardar los datos de la consulta
            $result = $this -> connection -> query($sql_statement);

            return $result;
```

Figura 1. Se muestra como insertar y consultar a los usuarios alumnos.

En la **Figura 2** se muestra un método que hace consulta por ID este busca y devuelve los datos de un alumno por su ID, el siguiente método actualiza la información de un alumno por su ID y el último método elimina al usuario por su ID.

```php
public function consultarPorID($id_browser){
    $sql_statement = "SELECT * FROM alumno WHERE idAlumno = ?";

    $statement = $this -> connection -> prepare($sql_statement);
    $statement -> bind_param("i", $id_browser);

    $statement -> execute();

    $result = $statement -> get_result();
    return $result -> fetch_assoc();
}

public function actualizarUsuario($idAlumno, $nombre, $apellidoP, $apellidoM, $usuario){
    $sql_statement = "UPDATE alumno SET nombre = ?, apellidoP= ?, apellidoM = ?,
     usuario= ? WHERE idAlumno = ?";

    $statement = $this -> connection -> prepare($sql_statement);
    $statement -> bind_param("ssssi", $nombre, $apellidoP, $apellidoM, $usuario, $idAlumno);

    return $statement -> execute();
}


public function eliminarUsuario($id_browser){
    $sql_statement = "DELETE FROM alumno WHERE idAlumno = ?";

    $statement = $this -> connection -> prepare($sql_statement);
    $statement -> bind_param("i", $id_browser);

    return $statement -> execute();
}
```

Figura 2. Se muestra cómo consultar, actualizar y eliminar a un usuario alumno por su ID.

Controlador

En la Figura 3 del controlador UserController.php se observa la inclusión del modelo y la conexión a la base de datos, la creación de la clase controladora con su constructor y el método insertUsuario(), el cual obtiene los datos del formulario de registro, encripta la contraseña del usuario con password_hash() para mayor seguridad y luego llama al modelo para insertar la información en la base de datos y finalizando con la carga de la vista registro_alumno.php.

```php
<?php

    // Incluir el modelo y la conexion a la BD
    include_once "app/models/UserModel_alum.php";
    include_once "config/db_connection.php";


    // Clase de controlador
    class UserController{

        private $model;

        // Constructor de la clase
        public function __construct($connection){

            $this -> model = new UserModel($connection);

        }

        // Metodo para obtener la infornacion del formulario
        public function insertUsuario(){
            if(isset($_POST['enviar'])){
                $nombre = $_POST['nombre'];
                $apellidoP = $_POST['apellidoP'];
                $apellidoM = $_POST['apellidoM'];
                $usuario = $_POST['usuario'];
                $correo = $_POST['correo'];
                $pass = password_hash($_POST['pass'], PASSWORD_BCRYPT);
                $genero = $_POST['genero'];

                // Llamar al metodo del modelo
                $insert = $this -> model -> insertarUsuario($nombre, $apellidoP, $apellidoM, $usuario
                , $correo, $pass, $genero);
            }
            //Incluir la vista
            include_once "app/views/registro_alumno.php";
        }
```

Figura 3. Se muestra controlador, constructor, insert y carga del registro.

En la **Figura 4** del controlador se muestran los métodos consultarUsuarios() y actualizarUsuario(). El primero obtiene todos los registros de usuarios desde el modelo y los envía a la vista consulta_alumno.php para mostrarlos. El segundo permite editar la información del alumno: primero recupera los datos existentes según el ID recibido por $_GET, y luego, si se envía el formulario por POST, actualiza los datos en la base de datos mediante el modelo y redirige según el resultado, utiliza la vista editar_alumno.php para mostrar el formulario de edición.

```php
public function consultarUsuarios(){
    $usuarios = $this -> model -> consultarUsuarios();

    include "app/views/consulta_alumno.php";

}

public function actualizarUsuario(){
    if(isset($_GET['id']) && is_numeric($_GET['id'])){
        $id_browser = (int) $_GET['id'];
        $row = $this -> model -> consultarPorID($id_browser);
        include_once "app/views/editar_alumno.php";
        return;
    }

    if(isset($_POST['editar'])){
        $idAlumno = (int) $_POST['id'];
        $nombre = trim($_POST['nombre']);
        $apellidoP = $_POST['apellidoP'];
        $apellidoM = $_POST['apellidoM'];
        $usuario = $_POST['usuario'];

        $update = $this -> model -> actualizarUsuario($idAlumno, $nombre, $apellidoP, $apellidoM,
        $usuario);

        if($update){
            header('Location: index.php?action=consult');
        }else{
            header('Location: index.php?action=update');
        }
    }
    include_once "app/views/editar_alumno.php";
}
```

Figura 4. Se muestran los métodos para consultar y actualizar datos.

En la **Figura 5** se muestra el método eliminarUsuario(), el cual se encarga de eliminar un registro de usuario de la base de datos. Primero verifica que se haya recibido un parámetro id válido mediante $_GET, y luego llama al método eliminarUsuario() del modelo para realizar la eliminación. Si la operación es exitosa, redirige a la vista de consulta (action=consult), y en caso contrario, redirige a una acción alternativa (action = delete).

```php
    public function eliminarUsuario(){
        if(isset($_GET['id']) && is_numeric($_GET['id'])){
            $id_browser = (int) $_GET['id'];

            $delete = $this -> model -> eliminarUsuario($id_browser);

            if($delete){
                header('Location: index.php?action=consult');
            }else{
                header('Location: index.php?action=delate');
            }
        }
    }
}
```

Figura 5. Método para eliminar.

Rutas

En la **Figura 6** se muestra el archivo principal que se encarga de gestionar las rutas del sistema. Aquí se incluye el controlador UserController_alum junto con las conexiones a la base de datos. Luego se crea una instancia del controlador y mediante la variable $_GET[‘action’] se identifica la acción solicitada en la URL. A través de un Switch, el sistema ejecuta el método correspondiente del controlador, como insertUsuario(), consultarUsuario(), actualizarUsuario() o eliminarUsuario(), permitiendo así manejar las operaciones y cambiar entre vistas según la acción.

```php
<?php

    //Llamar al controlador y a la conexión
    include_once "app/controllers/UserController_alum.php";
    include_once "app/controllers/UserController_prof.php";
    include_once "app/controllers/UserController_adm.php";
    include_once "config/db_connection.php";

    $controller = new UserController($connection);
    $controller_prof = new UserController_prof($connection);
    $controller_adm = new UserController_adm($connection);

    if(isset($_GET['action'])){
        $action = $_GET['action'];

        switch($action){
            case 'insert':
                $controller -> insertUsuario();
                break;
            case 'consult':
                $controller -> consultarUsuarios();
                break;
            case 'update':
                $controller -> actualizarUsuario();
                break;
            case 'delete':
                $controller -> eliminarUsuario();
                break;
```

Figura 6. Gestionador de rutas.

Conexión a la base de datos

En la **Figura 7** se muestra la conexión a la base de datos y el contenido de la tabla se muestran en la **Figura 10.**

```php
<?php
    // Crear las variables del servidor
    $server = "localhost";
    $user = "root";
    $password = "";
    $db = "repositorios";

    // Fncion para la conexion a la BD
    $connection = new mysqli($server, $user, $password, $db);

    if($connection -> connect_errno){
        // Conexion fallida
        die("Error en la conexion" . $connection -> connect_errno);
    }
```

Figura 7. Conexión a la base de datos.

Pruebas

Se presenta el formulario del registro alumno así como los campos a llenar con los datos del usuario como se muestra en la **Figura 8** y **Figura 9**.

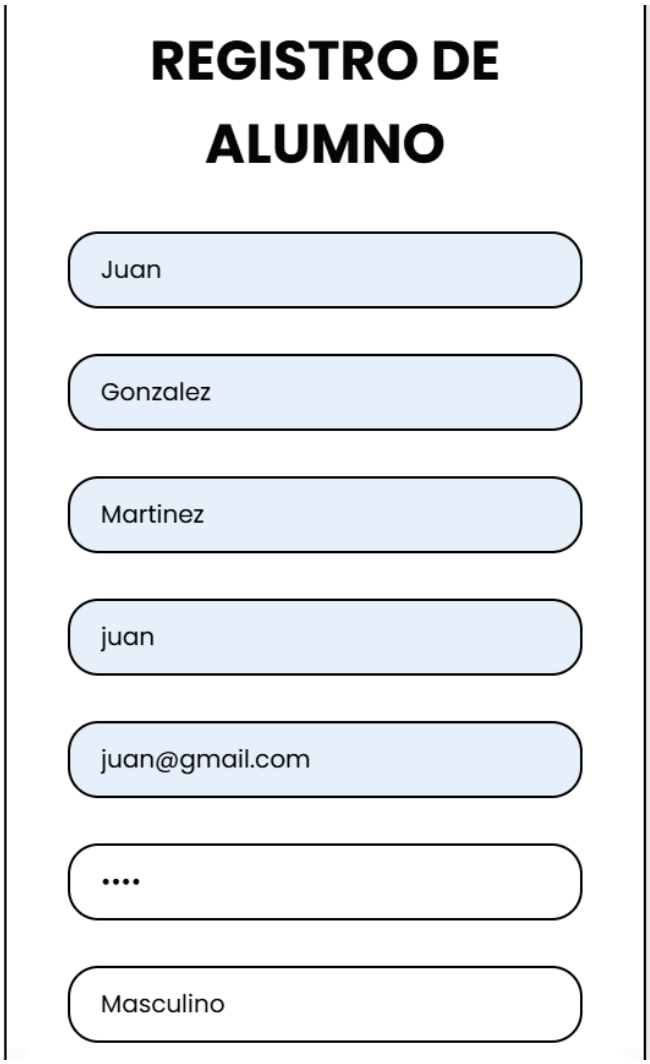

Figura 8. Se muestra el formulario registro alumno

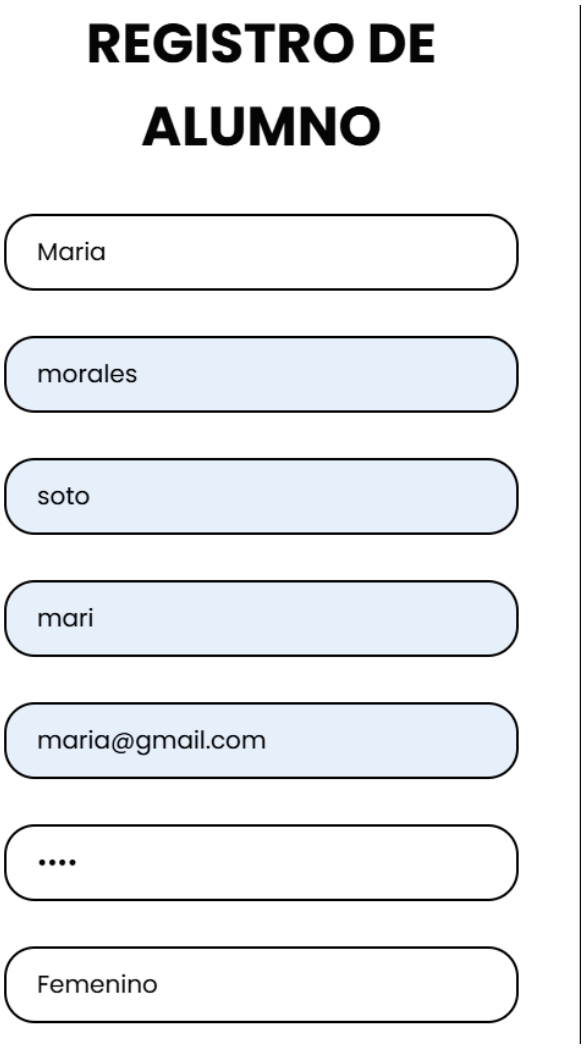

Figura 9. Se muestra el formulario registro alumno

En la **Figura 10** se muestra como los datos fueron capturados exitosamente en la base de datos repositorios en la tabla alumno.

idAlumno	nombre	apellidoP	apellidoM	usuario	correo	pass	genero
1	Juan	gonza	mar	pepito	juan@gmail.com	$2y$10$VT6iwm8zwprJGQlpO40pGupSQDA7zIL5c7ffHyeWquR...	Masculino
18	Juan	Gonzalez	Martinez	juan	juan@gmail.com	$2y$10$v8mRK.ve3sd7KCswfD1izeTvmXvcvyzteUPNjEtDr3F...	Masculino
19	Maria	morales	soto	mari	maria@gmail.com	$2y$10$0oc/dXdZGfiUA3pwqbXbv.6IKqkYeQvlQ8XeRe1fij2...	Femenino

Figura 10. Inserción en la base de datos tabla alumno.

Se realiza la consulta de los registros realizados anteriormente mostrando los datos exceptuando las contraseñas.

CONSULTA

ID	NOMBRE	APELLIDO PARTERNO	APELLIDO MATERNO	USUARIO	CORREO	GENERO	ACCIONES
1	Juan	gonza	mar	pepito	juan@gmail.com	Masculino	Editar Eliminar
18	Juan	Gonzalez	Martinez	juan	juan@gmail.com	Masculino	Editar Eliminar
19	Maria	morales	soto	mari	maria@gmail.com	Femenino	Editar Eliminar

Figura 11. Consulta los registros de la tabla alumnos

Se muestra como se edita al alumno y el botón para hacerlo **Figura 12,** en la **Figura 13** se muestra como los datos fueron actualizados correctamente.

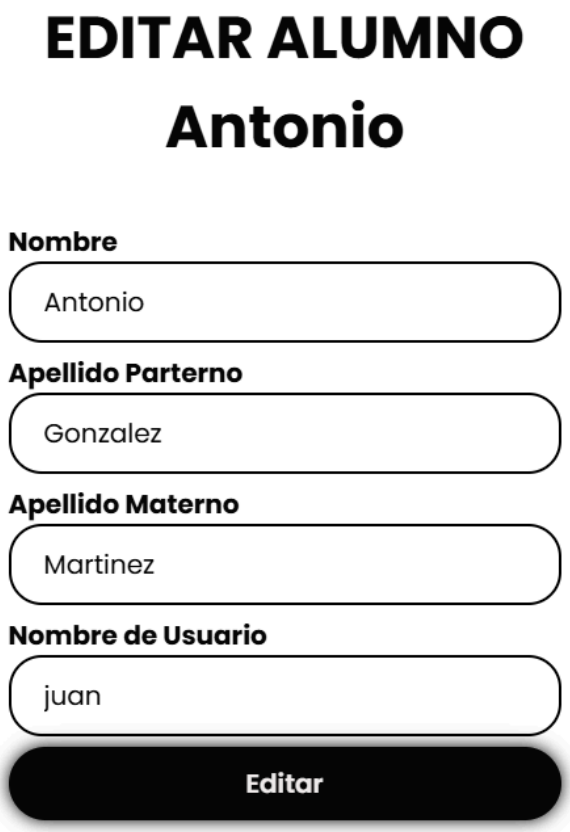

Figura 12. Formulario para editar y actualizar los datos de un alumno

ID	NOMBRE	APELLIDO PARTERNO	APELLIDO MATERNO	USUARIO	CORREO	GENERO
1	Juan	gonza	mar	pepito	juan@gmail.com	Masculino
18	Antonio	Gonzalez	Martinez	juan	juan@gmail.com	Masculino
19	Maria	morales	soto	mari	maria@gmail.com	Femenino

Figura 13. Se muestra la actualizacion del usuario

Por último hacemos la eliminación de un usuario con la confirmación de eliminar al usuario.

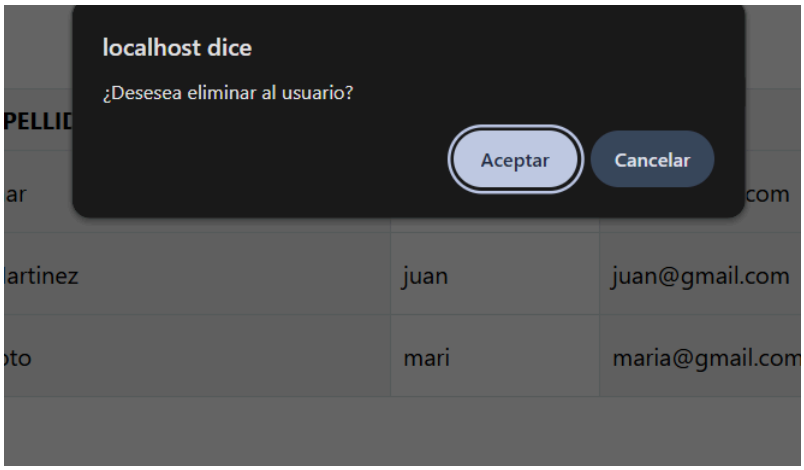

Figura 14. Confirmacion de eliminacion

ID	NOMBRE	APELLIDO PARTERNO	APELLIDO MATERNO	USUARIO	CORREO	GENERO
1	Juan	gonza	mar	pepito	juan@gmail.com	Masculino
18	Antonio	Gonzalez	Martinez	juan	juan@gmail.com	Masculino

Figura 15. Eliminación exitosa

Conclusión

En esta evidencia de producto se tuvieron complicaciones al realizar el editar y actualizar al usuario sin embargo se pudo resolver correctamente y con los aprendizajes que se obtuvieron durante las sesiones de clase. También aprendimos a usar la arquitectura MVC para un CRUD.