¿Que es MVC?:

Es un patrón de arquitectura de software que busca desacoplar la lógica en 3 capas:

- El controlador se encarga de recibir las peticiones que manda la vista.
- El modelo es el que define las reglas de negocio en nuestro caso las consultas a MySQL.
- La vista es lo que va a ver el usuario que en nuestro caso la maquetación en HTML.

Basándonos en nuestro proyecto anterior CRUD no mvc hemos pasado toda la lógica a MVC. En la cual hemos separado por completo el código HTMl del lenguaje PHP implementando un controlador, modelo y vista.

Distribución de nuestras carpetas

- Controller, contendra a todos nuestros controladores. Este esta compuesto por acciones, que en código son métodos/funciones.
- Model, contendra a nuestros modelos y entidades de negocio
- View, contendra a nuestras vistas (código HTML)
- index.php, sera nuestro FrontController

FrontController (index.php)

Nuestro proyecto comienzan ejecutando el archivo index.php, que va a hacer el papel de un FrontController, este es el encargado de saber que controlador y acción se esta ejecutando mientras que nuestro usuario navega. Nuestra lógica dice que mediante la queryString habran 2 parametros escenciales para saber que controlador y acción ejecutar el cual es "c" de controlador y "a" de acciòn.

```php
<?php
require_once 'model/database.php';

$controller = 'alumno';

// Todo esta lógica hara el papel de un FrontController
if(!isset($_REQUEST['c']))
{
    require_once "controller/$controller.controller.php";
    $controller = ucwords($controller) . 'Controller';
    $controller = new $controller;
    $controller->Index();
}
else
{
    // Obtenemos el controlador que queremos cargar
    $controller = strtolower($_REQUEST['c']);
    $accion = isset($_REQUEST['a']) ? $_REQUEST['a'] : 'Index';

    // Instanciamos el controlador
    require_once "controller/$controller.controller.php";
    $controller = ucwords($controller) . 'Controller';
    $controller = new $controller;
```

```
    // Llama la accion
    call_user_func( array( $controller, $accion ) );
}
```

Controlador

Ahora debemos crear **nuestras acciones** que en nuestro caso mostrara **diferentes vistas** o **acciones a realizar** como guardar/insertar/eliminar.

```php
<?php
require_once 'model/alumno.php';

class AlumnoController{

    private $model;

    public function __CONSTRUCT(){
        $this->model = new Alumno();
    }

    public function Index(){
        require_once 'view/header.php';
        require_once 'view/alumno/alumno.php';
        require_once 'view/footer.php';
    }

    public function Crud(){
        $alm = new Alumno();

        if(isset($_REQUEST['id'])){
            $alm = $this->model->Obtener($_REQUEST['id']);
        }

        require_once 'view/header.php';
        require_once 'view/alumno/alumno-editar.php';
        require_once 'view/footer.php';
    }

    public function Guardar(){
        $alm = new Alumno();

        $alm->id = $_REQUEST['id'];
        $alm->Nombre = $_REQUEST['Nombre'];
        $alm->Apellido = $_REQUEST['Apellido'];
        $alm->Correo = $_REQUEST['Correo'];
        $alm->Sexo = $_REQUEST['Sexo'];
        $alm->FechaNacimiento = $_REQUEST['FechaNacimiento'];

        $alm->id > 0
```

```
            ? $this->model->Actualizar($alm)
            : $this->model->Registrar($alm);

        header('Location: index.php');
    }

    public function Eliminar(){
        $this->model->Eliminar($_REQUEST['id']);
        header('Location: index.php');
    }
}
```

Los métodos Actualizar/Registrar/Eliminar de nuestro Controlador

Estos se encargan de comunicarse con el modelo para procesar la data contra nuestro base de datos de MySQL. Revisen el archivo alumno.model.php y encontaran los métodos que hacen las consultas.

Vista

Esta contendra el código HTML junto a algo de PHP para mostrarle al usuario sus interfaces de navegación. En nuestro caso hemos creado 2 archivos, uno llamado alumno.php que usaremos para listar la data en una tabla html y alumno-editar.php para crear/actualizar alumnos.

Vista > alumno.php (usado para listar data)

Si se dan cuenta, en los enlaces de editar y eliminar, apuntan a diferentes acciones del controlador Alumno.

- **Editar**. Le dice que se vaya a la acción Crud de nuestro clase AlumnoController que es nuestro controlador.
- **Eliminar**. Le dice que se vaya a la acción Eliminar de nuestro clase AlumnoController que es nuestro controlador.

```
<h1 class="page-header">Alumnos</h1>

<div class="well well-sm text-right">
    <a class="btn btn-primary" href="?c=Alumno&a=Crud">
      Nuevo alumno</a>
</div>

<table class="table table-striped">
    <thead>
        <tr>
            <th >Nombre</th>
            <th>Apellido</th>
            <th>Correo</th>
            <th >Sexo</th>
            <th >Nacimiento</th>
            <th ></th>
            <th ></th>
        </tr>
    </thead>
```

```
    <tbody>
    <?php foreach($this->model->Listar() as $r): ?>
        <tr>
            <td><?php echo $r->Nombre; ?></td>
            <td><?php echo $r->Apellido; ?></td>
            <td><?php echo $r->Correo; ?></td>
            <td><?php echo $r->Sexo == 1 ? 'Hombre' : 'Mujer'; ?></td>
            <td><?php echo $r->FechaNacimiento; ?></td>
            <td>
                <a href="?c=Alumno&a=Crud&id=<?php echo $r->id; ?>">
                  Editar</a>
            </td>
            <td>
                <a href="?c=Alumno&a=Eliminar&id=<?php echo $r->id;?>">
                  Eliminar
                </a>
            </td>
        </tr>
    <?php endforeach; ?>
    </tbody>
</table>
```

Alumno > alumno-editar.php (actualiza/registra nuevos alumnos). Nuestro formulario tiene un campo oculto (hidden) que almacena el ID del alumno, este nos va a servir para que el controlador sepa si tiene que registrar o actualizar al alumno.

```
<h1 class="page-header">
    <?php echo $alm->id != null ? $alm->Nombre : 'Nuevo Registro'; ?>
</h1>

<ol class="breadcrumb">
  <li><a href="?c=Alumno">Alumnos</a></li>
  <li class="active">
    <?php echo $alm->id != null ? $alm->Nombre : 'Nuevo Registro'; ?>
  </li>
</ol>

<form action="?c=Alumno&a=Guardar" method="post"
      enctype="multipart/form-data">
    <input type="hidden" name="id" value="<?php echo $alm->id; ?>" />

    <div class="form-group">
        <label>Nombre</label>
        <input type="text" name="Nombre"
               value="<?php echo $alm->Nombre; ?>"
               class="form-control" placeholder="Ingrese su nombre"
               data-validacion-tipo="requerido|min:3" />
    </div>
```

```
    <div class="form-group">
        <label>Apellido</label>
        <input type="text" name="Apellido"
               value="<?php echo $alm->Apellido; ?>"
               class="form-control" placeholder="Ingrese su apellido"
               data-validacion-tipo="requerido|min:10" />
    </div>

    <div class="form-group">
        <label>Correo</label>
        <input type="text" name="Correo"
               value="<?php echo $alm->Correo; ?>"
               class="form-control"
               placeholder="Ingrese su correo electrónico"
               data-validacion-tipo="requerido|email" />
    </div>

    <div class="form-group">
        <label>Sexo</label>
        <select name="Sexo" class="form-control">
            <option <?php echo $alm->Sexo == 1 ? 'selected' : ''; ?>
            value="1">Masculino</option> <option <?php echo $alm->Sexo
            == 2 ? 'selected' : ''; ?> value="2">Femenino</option>
        </select>
    </div>

    <div class="form-group">
        <label>Fecha de nacimiento</label> <input readonly type="text"
        name="FechaNacimiento" value="<?php echo
        $alm->FechaNacimiento; ?>" class="form-control datepicker"
        placeholder="Ingrese su fecha de nacimiento"
        data-validacion-tipo="requerido" />
    </div>

    <hr />

    <div class="text-right">
        <button class="btn btn-success">Guardar</button>
    </div>
</form>

<script>
    $(document).ready(function(){
        $("#frm-alumno").submit(function(){
            return $(this).validate();
        });
    })
</script>
```

Vista > header.php y footer.php

En MVC existe el concepto de layout, para no tener que agregar el código html de la cabecera y footer los creamos por separados y desde nuestro controlador los invocamos de la isguiente manera.

```
public function Index(){
    require_once 'view/header.php';
    require_once 'view/alumno/alumno.php';
    require_once 'view/footer.php';
}
```

Con esto evitamos tener que duplicar código haciendo nuestra aplicación más escalable

Tarea

1. Completar el CRUD

A partir de la rama «colegio» del repositorio actual: Estudiar qué falta, insertar, borrar, editar e incluirlo, si falta algo.

Terminar las entidades que hay en la base de datos: Curso y Curso_alumno (matrícula)

Añadir el controlador correspondiente, la tabla de la base de datos necesaria y las acciones correspondientes (CRUD).

Implementar un login.

Crear tabla de usuario en la base de datos. Antes de poder añadir, borrar, editar, se tiene que estar conectado (logueado).

Añadir cookies

Crear una sesión y añadir una cookie

Abrir una sesión, guardar una cookie que de la bienvenida a todo usuario conectado que vuelve a la página.