

Agenda en MVC



ÍNDICE

Introducción	3
Estructura	4
Base de datos	4
MVC	6
Realización	10
Clases	10
Validación de datos	11
Controladores	12
Funcionamiento	13
Conclusión	16

Introducción

La práctica consiste en la realización de una agenda siguiendo el modelo MVC (modelo - vista - controlador). En este informe se redacta la estructura del programa, así como su realización, funcionamiento y dificultades encontradas a lo largo del desarrollo del proyecto.

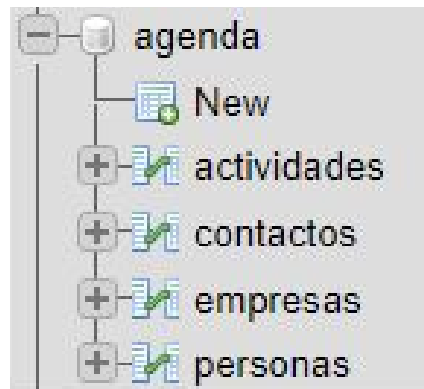
Estructura

En la agenda se pueden registrar dos tipos de contactos: personas y empresas. Todos los contactos almacenarán la misma información (ID, tipo, nombre y teléfono), y por separado, los contactos del tipo persona guardarán también la fecha del cumpleaños, mientras que los de tipo empresa, la página web de esta.

A parte de los contactos, también se pueden guardar las actividades de estos. La información almacenada es el título de la actividad (también será el ID de esta), la hora, fecha, lugar y la descripción de la misma.

Base de datos

Para ello, se ha creado la base de datos 'Agenda' siguiendo la estructura anteriormente mencionada:



Las tablas que la componen son las siguientes:


- Contactos

Como bien se mencionó antes, en la tabla de contactos, se almacena el ID del usuario, el tipo de contacto (persona o empresa), el nombre y el teléfono. La clave primaria es el ID.

#	Name	Type
<input type="checkbox"/> 1	id 	varchar(20)
<input type="checkbox"/> 2	tipo	varchar(8)
<input type="checkbox"/> 3	nombre	varchar(30)
<input type="checkbox"/> 4	telefono	varchar(11)


- Personas

Aquí se almacena el ID de la persona (DNI) y su fecha de cumpleaños. El ID es la clave primaria, siendo a su vez, una clave foránea de la tabla de contactos.

#	Name	Type
<input type="checkbox"/> 1	id 	varchar(20)
<input type="checkbox"/> 2	cumpleanos	varchar(50)

- Empresas

Similar a la tabla de personas: se almacena el ID de la empresa (CIAL) y la url de su página web. La clave primaria también es el ID, siendo clave foránea de la tabla contactos.

#	Name	Type
<input type="checkbox"/> 1	id 	varchar(20)
<input type="checkbox"/> 2	web	varchar(50)

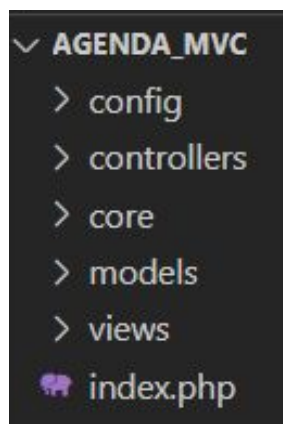
- Actividades

En esta tabla se almacena el título de la actividad, el ID del contacto con el que se realizará, y la hora, fecha, lugar y descripción de dicha actividad. La clave primaria es el título (ID), mientras que el ID del contacto es clave foránea de la tabla contactos.

#	Name	Type
<input type="checkbox"/> 1	id 	varchar(20)
<input type="checkbox"/> 2	idContacto 	varchar(20)
<input type="checkbox"/> 3	hora	varchar(5)
<input type="checkbox"/> 4	fecha	varchar(5)
<input type="checkbox"/> 5	lugar	varchar(50)
<input type="checkbox"/> 6	actividad	varchar(200)

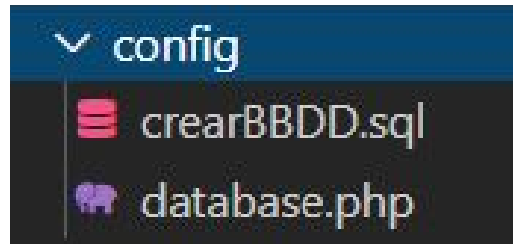
MVC

La organización del proyecto es en base al modelo MVC. Para ello la estructura de archivos debe lucir de la siguiente manera:



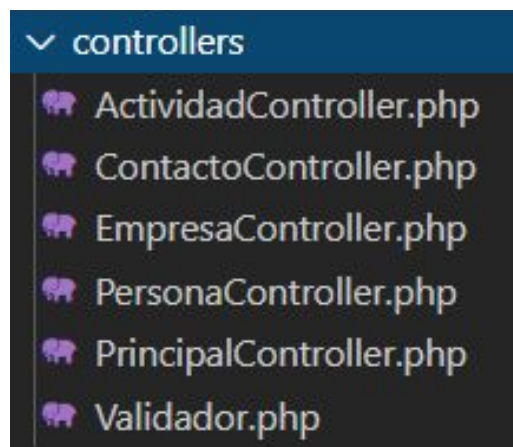
- Carpeta 'config'

En esta carpeta se encuentran dos archivos, ambos referentes a la base de datos. Uno de ellos, 'crearBBDD.sql' contiene el código necesario para poder crear la base de datos. El otro archivo 'database.php' tiene variables constantes con información necesaria para la creación de la base de datos.



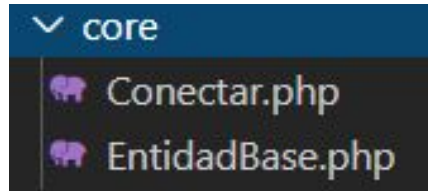
- Carpeta 'controllers'

En este directorio están todos los archivos controladores y un objeto 'trait' Validador, en este se encuentran métodos necesarios para la validación de datos. El 'PrincipalController.php' es el controlador principal, este se encarga de llamar a los otros controladores, dependiendo del tipo de dato que se vaya a manipular.



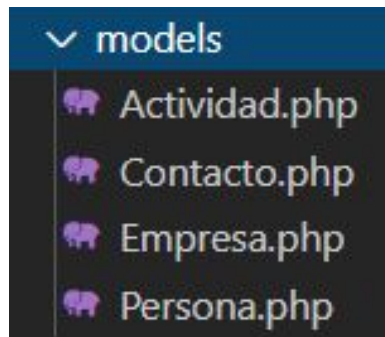
- Carpeta 'core'

En esta carpeta deben estar las clases “plantilla” de las que heredarán las demás. 'Conectar.php' contiene el código para crear la conexión con la base de datos, mientras que en 'EntidadBase.php' están algunos métodos por defecto que todas las demás clases podrán usar.



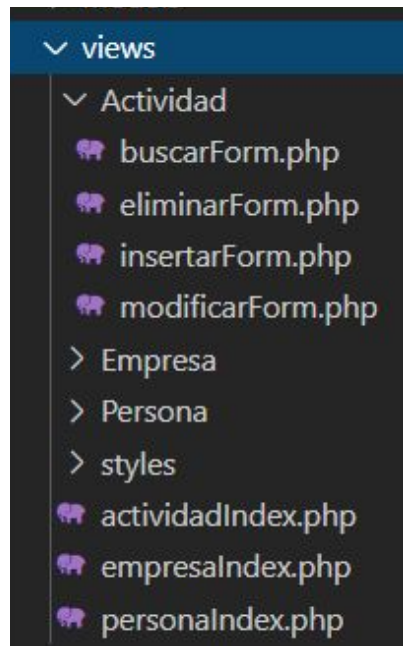
- Carpeta 'models'

Aquí están todas las clases necesarias para la realización del proyecto. Todas las clases tienen métodos heredados de 'EntidadBase' para poder manipular los datos, así como también tienen los suyos propios.



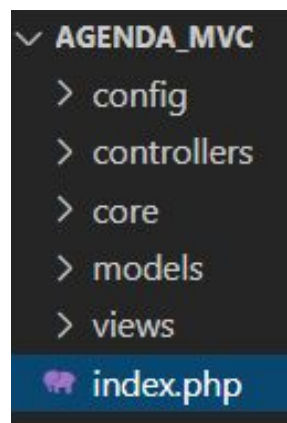
- Carpeta 'views'

En esta carpeta están todos los archivos referentes a lo que el usuario ve y con lo que interactúa. Las carpetas 'Actividad', 'Empresa' y 'Persona' contienen las vistas de los formularios correspondientes a cada uno. En el directorio 'styles' se encuentra un archivo 'css' donde están los estilos de toda la página.



- Archivo 'index.php'

Este archivo es la página principal de todo el proyecto.



Realización

Clases

Para la implementación de las tablas en código, se ha hecho el uso de clases.

- Clase 'Conectar'
Esta clase crea la conexión con la base de datos para poder manipularla.
- Clase 'EntidadBase'
Esta clase es la "plantilla" a seguir de las demás. En su constructor se crea la conexión con la base de datos con el objeto 'Conectar'. A parte de los 'getters' y 'setters', tiene otros métodos:
 - update(\$id, \$tabla, \$columna, \$valor): realiza la actualización de la columna indicada al valor indicado y devuelve si se ha hecho con éxito o no.
 - getAll(\$sql): obtiene todos los datos de la tabla indicada en la sentencia 'sql'. Se recibe el 'sql' como parámetro pues hay ocasiones en las que es necesario realizar 'joins' entre tablas para obtener los datos solicitados.
- Clase abstracta 'Contacto'
De esta clase heredan las otras dos 'Empresa' y 'Persona', pues ambas tienen atributos en común. Esta clase hereda de 'EntidadBase', por lo que 'Empresa' y 'Persona' también pueden hacer uso de sus métodos.
- Clases 'Empresa' y 'Persona'
Estas dos clases heredan de 'Contacto', como se mencionó antes, por lo que solo se les declara como atributo 'web' y 'cumpleanos', respectivamente. Con estas clases es necesario manipular dos tablas, la de 'contactos' y la otra correspondiente. Es por esto que los métodos de insertar, eliminar y buscar por ID no se encuentran en 'EntidadBase'.

- insert(): se ejecutan las sentencias 'sql' correspondientes para insertar los datos en la base de datos y devuelve si se realizó con éxito o no.
 - delete(\$id): se elimina de ambas tablas ('empresas' / 'personas' y 'contactos') el contacto cuya ID es la recibida como parámetro.
 - getByID(\$id): se obtienen los datos de ambas tablas del contacto indicado por parámetro y los devuelve si lo encontró, si no se devuelve un '0'.
- Clase 'Actividad'
Esta clase hereda de 'EntidadBase'. Sus métodos son los mismos que en 'Empresa' y 'Persona', pero adaptados para su correcto funcionamiento.

Validación de datos

Para asegurarnos de que los datos introducidos por el usuario son correctos, se ha creado un 'trait' llamado 'Validador'. Las clases de los controladores lo implementan y utilizan sus métodos para hacer las comprobaciones necesarias.

- soloNumeros(\$texto): comprueba si el texto enviado está compuesto solo por números.
- validarHora(\$hora): comprueba si la hora enviada está en el formato hh:mm.
- validarFecha(\$fecha): comprueba si la fecha enviada como parámetro está en el formato dd/mm.
- validarDni(\$dni): comprueba si el DNI enviado es real. (Para la inserción de personas es más cómodo usar un [generador de DNI](#)).
- validarCial(\$cial): comprueba si el CIAL enviado está compuesto por letras y números.

Controladores

En total hay cinco controladores, siendo uno de ellos el que se encarga de llamar a los otros, el principal.

- **PrincipalController**
Este es el que se encarga del funcionamiento de toda el programa. Según el tipo de acción que reciba en `$_REQUEST` enviada por los formularios, llamará al controlador correspondiente y realizará la acción solicitada.
- **PersonaController, EmpresaController y ActividadController**
Se encargan de validar los datos recibidos, ya sea por método POST o GET, y llamar al modelo correspondiente para interactuar con la base de datos.
- **ContactoController**
Este controlador está creado con el único fin de poder mostrar todos los contactos existentes independientemente de su tipo.

Funcionamiento

Para poder ver el proyecto en funcionamiento, es necesario tener la aplicación XAMPP o similar instalada. Iniciamos los servicios de Apache y MySQL y copiamos el script sql que está en la carpeta 'config' en el 'phpMyAdmin' para que así se cree la base de datos.

La carpeta del proyecto se pega en la carpeta htdocs que se encuentra en la carpeta del XAMPP y entramos en http://localhost/Agenda_MVC.



Para mostrar un ejemplo del funcionamiento voy a insertar una persona. Hacemos click en el botón de Persona o vamos al enlace de la barra de navegación. Seleccionamos la opción de insertar y rellenamos el formulario con los datos correctos e insertamos.

AGENDA Persona Empresa Actividad

Insertar persona

DNI

97247946J

Nombre

Alanis Simoes

Teléfono

653075911

Cumpleaños

23/06

Insertar

Según el resultado de la operación, nos aparecerá un mensaje indicando si se ha realizado correctamente la acción o no.

AGENDA Persona Empresa Actividad

**Se ha introducido correctamente a Alanis Simoes
en la agenda**

Si vamos a Persona desde la barra de navegación y pulsamos Listar personas nos aparece una lista con los contactos de tipo Persona en la agenda.

AGENDA Persona Empresa Actividad			
Lista personas			
DNI	Nombre	Teléfono	Cumpleaños
97247946J	Alanis Simoes	653075911	23/06

Desde la página de inicio, pulsamos el botón de listar contactos para ver la lista de todos los contactos, ya sea empresa o persona.

AGENDA Persona Empresa Actividad					
Lista contactos					
DNI/CIAL	Tipo	Nombre	Teléfono	Fecha de cumpleaños	Página web
97247946J	Persona	Alanis Simoes	653075911	23/06	-

Conclusión

Me surgieron varios problemas a lo largo de la realización de esta práctica, pues no me aclaraba con la estructura de la base de datos. También tuve que cambiar muchas veces la clase de 'EntidadBase' pues al tener que manipular datos en dos tablas al mismo tiempo tuve que quitar varios métodos de esa clase y meterlos en cada una y adaptarlo.

La práctica me pareció entretenida, aunque me hubiese gustado tener un poco más de tiempo para así poder pulir muchas partes del código que yo misma sé que son mejorables. De todas maneras, me ha ayudado a entender mejor cómo funciona el modelo de MVC.