



IES ZAIDÍN-VERGELES GRANADA

Ciclo de Grado Superior Desarrollo de Aplicaciones Web

“GROUP MANAGER, MODELO DE GESTIÓN DE GRUPOS”

Junio 2023

Trabajo realizado por Joaquín Melero Peralta

Dirigido por Abel Martos López

ÍNDICE DE CONTENIDOS

1. Definición del Proyecto. Resumen.....	3
1.1 Justificación del proyecto.	3
1.2. Objetivos de la app web.....	4
1.3. Explicación detallada del funcionamiento de la aplicación web (usuario).....	4
1.3.1 Pantalla de Inicio o Index de la aplicación web.	4
1.3.2 Contenido de la app web una vez autenticado.	5
A. Acceso master o administrador	5
B. Acceso departamento o coordinador equipo departamento.....	6
C. Acceso miembro de grupo o empleado	11
2. Desarrollo del proyecto.	12
2.1 Investigación y planificación	12
2.2 Desarrollo.....	13
2.2.1 Estructura de la app web	13
2.2.2 Lenguajes de programación, tecnologías utilizadas y librerías externas:	26
2.2.3 Base de datos.....	27
2.3 Diseño	28
2.3.1 Diseño Responsive o adaptativo	28
2.3.2 Estilos	29
3. Conclusión.....	30
3.1 Reflexión personal	30
3.2 Aprendizajes adquiridos	31
3.3 Vías futuras de desarrollo.	32
4. Bibliografía.....	33

1. Definición del Proyecto. Resumen.

Este proyecto parte desde cero, sin ninguna base, plantilla o framework básico. Consiste en el desarrollo de una aplicación web destinada a almacenar y gestionar de forma segura todos los datos relevantes de personas que pertenecen a un conjunto, ya sean empleados de una empresa, integrantes de un equipo de trabajo o determinados sujetos de un grupo y establecer una comunicación unilateral con ellas.

De esta manera, se pueden llevar a cabo las actividades administrativas y las estructuras organizativas de una estructura de forma fácil, rápida y segura. La aplicación consta de varias interfaces principales, cuyo aspecto y funcionalidad depende del rol del usuario registrado.

La aplicación se desarrolla en web, de forma que los usuarios podrán acceder a ella sin necesidad de instalar software en su equipo informático y con el único requisito de tener un navegador web y una conexión a internet.

Palabras clave: modelo de gestión de empleados, mvc, desarrollo web, app web, javascript, php, html, css, base de datos.

1.1 Justificación del proyecto.

Desde hace unos años, cada vez más empleados teletrabajan o tienen un entorno híbrido. Los departamentos de recursos humanos, equipos de trabajo o grupos puntuales de colaboración demandan cada vez más aplicaciones y herramientas web que les permitan realizar unas determinadas acciones y control sobre sus integrantes, desde cualquier parte y dispositivo, de forma rápida y sencilla.

Este proyecto nace de la idea básica de un CRM (Customer Relationship Management), con el objetivo de crear una app web para gestionar un conjunto de prácticas, estrategias comerciales y tecnologías enfocadas en la relación con el cliente. Pero finalmente se desmarca de este tipo de softwares y se centra en la gestión de individuos dentro de una organización, departamento, equipo o grupo.

1.2. Objetivos de la app web.

1. Agilizar las operaciones de administración.

Gracias a esta aplicación se puede acceder de forma rápida a toda la información de los miembros de un grupo. Se muestra una visión global de todos por defecto con la posibilidad de acceder a una vista específica de cada componente, donde se encuentran más detalles individuales. Planificar y programar el trabajo de todo el equipo y asignar tareas fácilmente a sus empleados.

2. Facilitar la toma de decisiones a la hora de asignar tareas y mejorar la comunicación interna.

Desde la vista de tareas, se puede llevar un seguimiento de las tareas pendientes por realizar y su correspondiente encargado de llevarla a cabo.

3. Llevar a cabo un control histórico de tareas realizadas.

Una vez marcada una tarea como realizada pasará a una tabla donde se registrará con la fecha de iniciación y fin de esta. Esta tabla se encuentra en la base de datos.

4. Diseñar una base de datos con la información de los miembros para tenerla organizada y centralizada.

1.3. Explicación detallada del funcionamiento de la aplicación web (usuario).

1.3.1 Pantalla de Inicio o Index de la aplicación web.

En esta vista, el usuario encuentra un login para iniciar sesión y/o un registro para crear un nuevo usuario.

En el proceso de registro se solicitan:

- Un campo de descripción del usuario
- Nombre de usuario
- Contraseña
- Selección tipo de usuario. Usuario departamento o empleado.

The image shows a login and registration form for a CRM system. The form is centered on a blue background with a blurred image of a person. At the top, there is a logo with a stylized 'C' and the text 'CRM JOAQUÍN MELERO'. Below the logo, the text 'Accede a tu cuenta' is displayed. Underneath, there are two input fields: 'Usuario' and 'Contraseña', followed by an 'Acceder' button. Below this, the text 'Registro de Nuevo Usuario' is shown. This section contains four input fields: 'Nuevo Nombre Usuario', 'Nuevo Usuario', 'Nueva Contraseña', and 'Selecciona' (with a dropdown arrow). A 'Registrar' button is positioned below these fields. At the bottom, there is a copyright notice '© 2023 Joaquín Melero'.

Figura 1. Login/Registro

1.3.2 Contenido de la app web una vez autenticado.

A. Acceso master o administrador: con esta autenticación tiene acceso a todos los usuarios creados en la aplicación y tiene permisos para editar o borrar cualquiera. Esta acreditación es única y no se puede acceder en el proceso de registro, se facilita por el desarrollador web.

El menú de navegación muestra:

- Listado de Empleados
- Cerrar Sesión
- Información sobre el usuario y sesión actual iniciada



Figura 2. Vista Administrador

B. Acceso departamento o coordinador equipo departamento: esta autenticación da acceso a la principal funcionalidad de la aplicación web.

En la vista inicial se muestra un menú superior de navegación en el que se muestran las siguientes opciones:

- Crear Empleado
- Listado de Empleados
- Tareas de Departamento
- Cerrar Sesión
- Información sobre el usuario y sesión actual iniciada

En el cuerpo de la pantalla se listan los empleados del departamento.

La parte inferior o footer muestra el logo y Copyright.



Figura 3. Vista Departamento Inicial

➤ Card o Tarjeta de empleado: Estas tarjetas muestran la información personal general de cada empleado y un menú de funcionalidades.

- Información de empleado:
 - Nombre y Apellidos
 - Nombre del departamento
 - Puesto
 - Información de empleado:
- Menú (iconos)
 - Redactar tarea
 - Abrir información personal del empleado
 - Notificaciones de las tareas pendientes de realizar

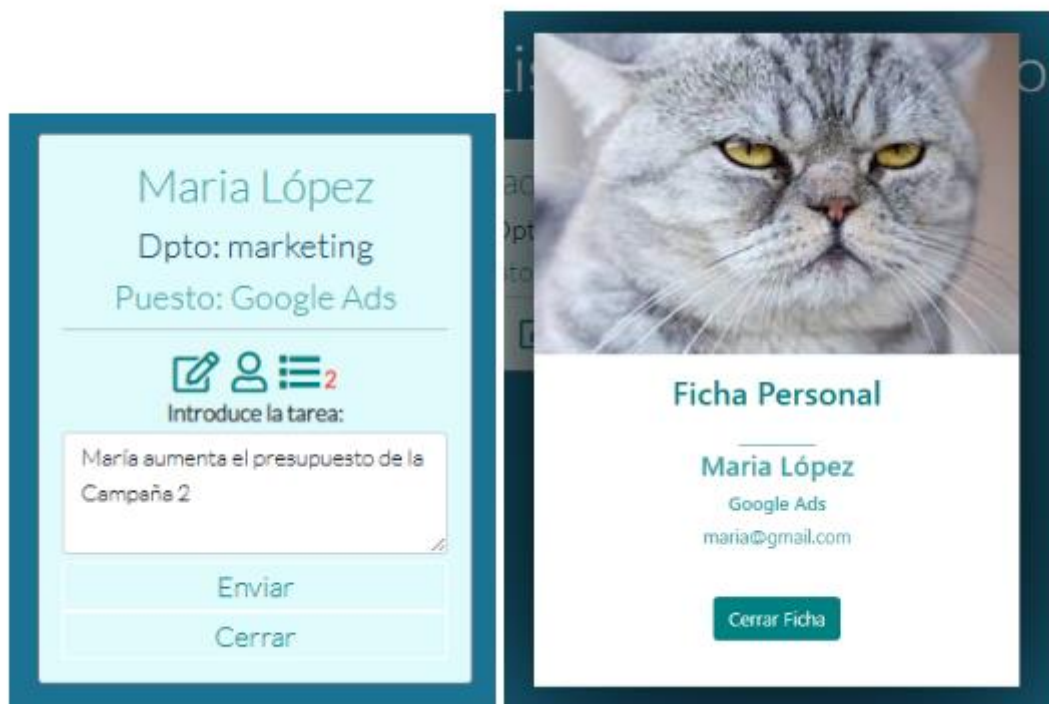


Figura 4 .Menú card

- Vista Tareas de Departamento: Para acceder a esta página se puede hacer mediante el menú superior en “Tarea de departamento” o directamente haciendo clic en el icono del menú de las tarjetas de los empleados, donde se ve también el número de tareas pendientes de cada uno.

Crear Empleado Listado de Empleados Tareas de Departamento **Cerrar Sesión** Sesión: marketing Dpto: marketing

Dpto. de marketing: Daily To Do Lists

Filtrar tarea por empleado:

Foto	Nombre	Mensaje	Estado
	Maria López	María prepara la campaña para el ciclo	Realizado: <input type="checkbox"/>
	Joaquín Melero	Envía el informe de ventas CD	Realizado: <input type="checkbox"/>

Historial | Cerrar Historial



 © 2023, Joaquín Melero

Figura 5. Vista Tareas de Departamento

La página se compone de los siguientes elementos:

1) Menú de navegación superior:

- Crear Empleado
- Listado de Empleados
- Tareas de Departamento
- Cerrar Sesión
- Información sobre el usuario y sesión actual iniciada

2) Sección de búsqueda de empleados por filtros:

- Selección de empleados activos
- Botón de filtrar para realizar búsqueda
- Botón de exportar tabla a excel (.csv)
- Botón reset para limpiar la búsqueda

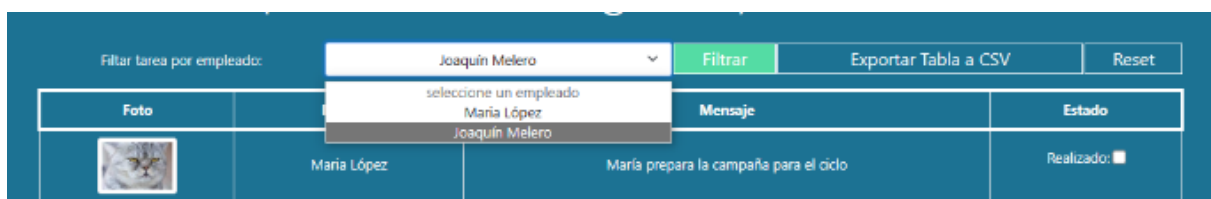


Figura 6. Sección búsqueda por empleado

3) Tabla para listar los empleados con tareas activas o pendientes:

- Foto: imagen en miniatura del empleado
- Nombre: nombre y apellidos del empleado
- Mensaje o tarea: contenido de la tarea a desarrollar
- Estado: checkbox para marcar cuando se ha realizado una tarea





Foto	Nombre	Mensaje	Estado
	Maria López	María prepara la campaña para el ciclo	Realizado: 
	Joaquín Melero	Envía el informe de ventas QD	 Checked

Figura 7. Tabla de tareas pendientes

4) Historial de tareas, nueva tabla que recoge las tareas ya realizadas. Solo se muestra al pulsar en enlace Historial, se compone de :

- Nombre: nombre y apellidos del empleado
- Mensaje o tarea: contenido de la tarea a desarrollar
- Fecha de Inicio: fecha y hora de cuando se encargó la tarea
- Fecha Fin: fecha y hora de cuando se finalizó la tarea

Historial Cerrar Historial			
Nombre	Tarea	Fecha Inicio	Fecha Fin
Maria López	Maria realiza la optimización 2	2023-05-12 09:14:14	2023-05-25 20:22:48
Juan Rodríguez	Juan prepara los informes del cliente B	2023-05-12 17:32:54	2023-05-15 13:50:36
Maria López	Retira la campaña del concesionario de Granada	2023-05-15 13:54:23	2023-05-15 13:54:27

Figura 8. Tabla de historial de tareas

- Vista Crear Empleado: para acceder a esta vista hacemos clic en el menú superior de navegación en “Crear Empleado”. Desde esta página podemos crear los miembros del grupo o empleados de departamento. Los sujetos que se crean pertenecen predeterminadamente al departamento (usuario) con el que se ha autenticado.

Estos son los campos requeridos para dar de alta un nuevo sujeto:

- Nombre: nombre y apellidos del empleado.
- Puesto: rol o desempeño que tiene el componente en el grupo
- Email: dirección de correo electrónico para contacto
- Imagen: foto del individuo

Como particularidad del proceso de creación de miembros, decir que, el campo “Nombre” es único, por lo que cuando se introduce, se realiza una comprobación

de no coincidencia, y si está libre se procede a habilitar el siguiente campo “Puesto”.

El procedimiento de alta se cierra con una validación de todos los campos, si es correcta se habilitará un botón de “Guardar”.



The screenshot shows a web application interface for creating a new employee. The header bar is dark blue with navigation links: 'Crear Empleado' (active), 'Listado de Empleados', 'Tareas de Departamento', 'Cerrar Sesión', and 'Sesión: marketing Dpto: marketing'. The main title 'Crear Empleado' is centered in white. Below it, there are four input fields: 'Nombre' (containing 'Lola Torres'), 'Puesto' (containing 'Influencer'), 'Email' (containing 'lola_torres@gmail.com'), and 'Subir imagen' (with a file selection button 'Elegir archivo' and the filename 'abeja.jpg'). At the bottom of the form are two buttons: 'Guardar' (highlighted in green) and 'Cancelar'. The footer contains the CRM logo, the name 'Joaquín Melero', and the copyright notice '© 2023, Joaquín Melero'.

Figura 9. Vista Crear Empleado

C. Acceso miembro de grupo o empleado: con esta autenticación se tiene acceso a una vista general para empleados o miembros del grupo. La funcionalidad de esta página es seleccionar el nombre del empleado deseado y listar en una tabla las tareas incompletas o pendientes. Para ganar claridad y tener una visión global de todos los empleados y tareas, se permite mostrar las tareas de tantos empleados o miembros se seleccione. Con el botón “Reset” se reiniciará la vista. Para salir o cerrar sesión basta con pulsar en el menú de navegación superior la opción “Cerrar sesión”.



Figura 10. Vista general con credencial empleado

2. Desarrollo del proyecto.

En este apartado se describe desde un punto de vista más técnico la funcionalidad y proceso de creación de la app web.

2.1 Investigación y planificación

Es el primer paso de un proyecto web. El desarrollo comienza con la fase de investigación y planificación. En esta etapa se decide desarrollar una aplicación web para la gestión de grupos o empleados y que se dirigirá a usuarios con la necesidad de gestionar o manejar grupos.

Después se consultan diferentes apps de gestión de empleados y de asignación de tareas y se determinan una serie de funcionalidades a llevar a cabo, además de la primera arquitectura web y la forma de navegar entre los contenidos, resumiendo:

- La app necesita un login que se encarga de la autenticación del usuario (comprobando que el nombre de usuario y contraseña sean correctos), y establece un entorno inicial para el usuario, dependiendo del tipo o rol.

- Todas las páginas deberán de contar un menú superior de navegación donde se indique:
 - Nombre de usuario con el que se ha autenticado el usuario que ha accedido y el grupo o departamento al que pertenece.
 - Cierre de sesión que te devuelve a la página de login y borra la sesión actual.
 - Dependiendo del rol de acceso se mostrarán unas funcionalidades u otras: Crear Empleado, Editar, Eliminar, Listar.
- Se creará una base de datos para almacenar los usuarios, empleados y tareas.

2.2 Desarrollo

2.2.1 Estructura de la app web

La aplicación web se estructurará siguiendo como base un patrón de diseño MVC. Modelo-vista-controlador, abreviado MVC, es un tipo de arquitectura distribuida de software en la que se estipula que los datos (modelo), la interfaz de usuario (vista) y la lógica de control (control) son tres componentes distintos. Cada uno de estos componentes desempeña funciones específicas.

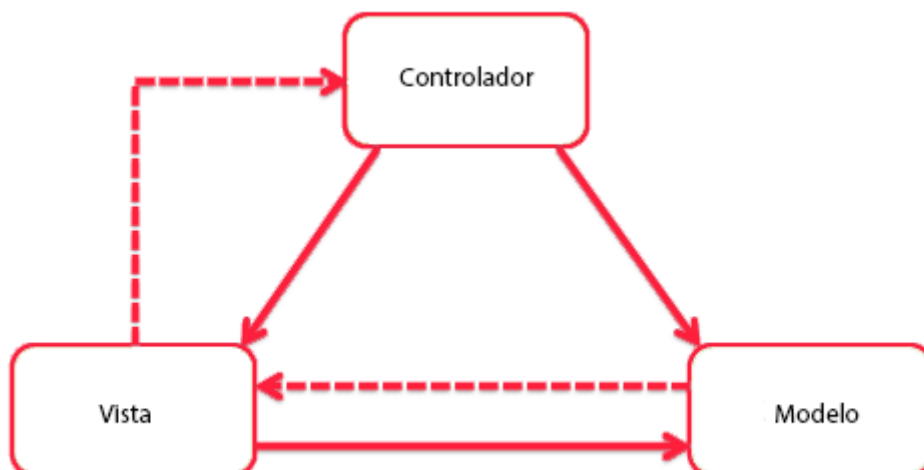


Figura 11. Esquema básico MVC

Este tipo de arquitectura de software es muy conocida y se lleva utilizando en todo tipo de aplicaciones, ya que ha demostrado con creces su validez tanto en aplicaciones web como en programas de escritorio.

El comportamiento, de forma resumida, sería el siguiente:

La vista es el componente encargado de generar la interfaz de usuario. El modelo se encarga de gestionar los datos y la información almacenada en la base de datos. El controlador podría decirse que es un intermediario entre los anteriores componentes, que se encarga de indicar a la vista y al modelo como deben actuar a continuación.

En este proyecto, sobre este modelo, se añade un nuevo componente de operabilidad. Este se encarga de añadir nuevas funciones al modelo, como el validar un determinado campo o mostrar una información determinada cuando el usuario hace una acción específica.

Distribución de carpetas que componen el proyecto:

Estos son los directorios principales que se encuentran en la raíz del sistema:

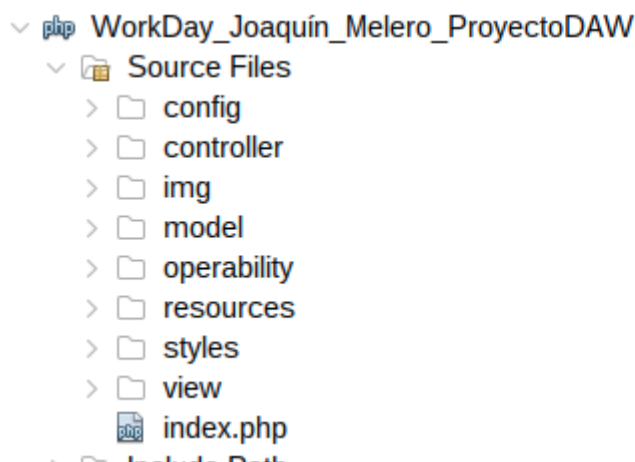


Figura 12. Vista estructura carpetas proyecto

- **Index:** En este fichero recibiremos todas las peticiones de los controladores.

La forma de acceder a la aplicación es llamar al archivo `index.php`, pasándole como parámetros el nombre del controlador y la acción que necesitamos. Será el

propio controlador el que decida que vista se va a cargar, según las que tengamos definidas.

Ejemplo: Acceso a vista lista de tareas ->
`../Workday_Joaquin_Melero_ProyectoDAW/index.php?controller=message_controller&action=inbox.`

En ese archivo también se llama a `session_start()` para iniciar la sesión del usuario.

- **Carpeta config:** se encuentra el archivo `config.php` se encuentra la información del controlador y la acción por defecto. Si se ha producido un inicio de sesión con un acceso autenticado se carga por defecto el controlador y vista de empleados, si no se ha producido esta autenticación, se carga por defecto el controlador que te lleva a la vista del login.
- **Carpeta controller:**

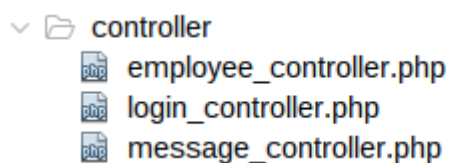


Figura 13. Vista del contenido carpeta controller

En este directorio encontramos los archivos “`employee_controller.php`”, “`login_controller.php`” y “`message_controller.php`”. Estos se encargan de solicitar datos al modelo y enviarlos a la vista. Interconectan el frontend con el backend. El código de cada archivo se organiza en clases:

→ `employee_controller.php`:

La clase `employee_controller` se encarga de procesar todo lo relacionado con los empleados (miembros del grupo) obteniendo la información de la base de datos a través de `employee model`. Está formada por:

- propiedades:
 - `$page_title` (guarda el título de la vista).

- \$view (guarda la vista).
- \$employeeObj (objeto de clase Empleado).
- constructor
 - inicializa las propiedades.
- métodos:
 - list(): lista los miembros de la tabla employees.
 - edit(\$id=null): recibe un id desconocido, se comprueba si se ha guardado un id y se carga la vista de editar o crear. Devuelve el registro con esa id.
 - save(): llama el método del model que guarda en la base de datos el registro.
 - confirmDelete(): carga la vista que pregunta si se desea eliminar el registro.
 - delete(): llama al método del model que elimina el registro de la base de datos.
 - deletejs(): método al que se accede desde ajax en archivo js para eliminar un registro con un id recibido pos POST.
 - getFicha(): método que obtiene del model los datos de un registro con una id dada.
 - checkNameEmployee(): método que comprueba si existe en la base de datos un nombre ya registrado.
 - upImg(): método para comprobar la imagen que el usuario sube y guardarla en la carpeta específica del proyecto.

→ login_controller.php:

Controlador que contiene la clase login_controller que tiene como finalidad interaccionar con el modelo login y la vista de login. Recibe los datos desde la vista login por ajax y se llama al método que comprueba que los datos recibidos por el form y los que están en la base de datos coinciden.

Está formada por:

- Propiedades:
 - \$view: almacena la vista.
 - \$loginObj: objeto login_model.
 - \$page_title: título de la página.
- Constructor:

Inicializa \$view y \$loginObj.
- Métodos:
 - login(): carga la vista de “Login.”
 - checkAcces(): comprueba las credenciales recibidas por ajax con las de la BD.
 - logOgg(): método que borra las sesiones y devuelve al login o index.
 - addRegister(): método para mostrar el formulario y mandar los datos del registro al modelo login donde los introduce en la base de datos.
 - getRolEmployee(): recibe un id de usuario, llama al método del modelo y devuelve el rol de ese usuario.

→ message_controller.php:

Controlador que contiene la clase para controlar/relacionar la vista del usuario de enviar mensajes y la base de datos donde se guardarán. Está formada por:

- Propiedades:
 - \$view: atributo para guardar la vista.
 - \$messageObj: lobjeto Message_model.
 - \$page_title: atributo para guardar el título.
- Constructor: Inicializa las propiedades view y messageObj.

- **Métodos:**
 - `inbox()`: Lista de mensajes o tareas.
 - `saveMessage()`: método que recibe del `message.js` los campos a registrar en la tabla mensajes en la base de datos.
 - `completeMessage()`: método que manda al modelo la id de un mensaje para poner la tarea realizada.
 - `deleteMessage()`: le pasa la id al método del modelo para borrar este mensaje en la base de datos.
 - `getFoto($id, $name)`: método que llama al modelo para obtener la foto de un id y su nombre.
 - `contTareasEmployee($nameEmployee)`: método que recibe `idUser` de un empleado y devuelve el número de tareas que tiene pendiente.
 - `listarTareasEmpleado()`: método que recibe el nombre de un empleado por ajax y devuelve el listado de tareas.

- **Carpeta `img`:** directorio del proyecto para guardar las imágenes subidas en la creación de un empleado.

- **Carpeta `model`:** en este directorio se encuentran todos los modelos, son los ficheros que contienen las clases encargadas de gestionar los datos y la información almacenada en la base de datos.

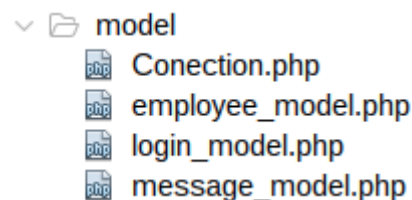


Figura 14. Vista del contenido carpeta model

→ Conection.php:

Modelo que contiene la clase `ConectionDB` para establecer la conexión con la base de datos. Está formada por el método estático `conection()` para

no crear instancia y acceder directamente a él. Devuelve true si se ha establecido la conexión correctamente.

→ employee_model.php:

Este fichero es el modelo principal. Contiene la clase Employee y sus métodos que operan para obtener la información en la base de datos. Está formada por:

- Propiedades:
 - \$table: nombre de la tabla de la base de datos..
 - \$connection: conexión a la base de datos.

- Métodos:
 - getConnection(): método que guarda en la propiedad \$connection la conexión que devuelve el método de la clase Connection.
 - getEmployees(): método que devuelve un array que contiene todas las filas restantes del conjunto de resultados.
 - getEmployeeById(\$id): método que devuelve devuelve un array que contiene todas las filas restantes del conjunto de resultados.
 - save(\$param, \$urlFoto): método que guarda un empleado en la base de datos. Comprueba si ya existe y lo actualiza, y si no, lo crea en la base de datos.
 - deleteEmployeeById(\$id): método que recibe la id de un empleado y elimina a este y sus tareas pendientes de la base de datos.

→ login_model.php:

Este fichero es el modelo principal. Contiene la clase Employee y sus métodos que operan para obtener la información en la base de datos. Está formada por:

- Propiedades:

- \$table: nombre de la tabla de la base de datos.
- \$connection: conexión a la base de datos.

- Métodos:

- getConnection(): instancia al objeto Connection e inicializa la propiedad \$connection.
- getUserPassDB(\$username): método que comprueba en la tabla user de la base de datos si los usuarios están registrados y los devuelve si es así.
- addUser(\$names, \$username, \$pass, \$rol): método que realiza una consulta “insert” con los parámetros recibidos en la base de datos.
- getRol(\$idUser): método que recibe el id de un usuario y devuelve el rol de este consultando en la base de datos.

→ message_model.php:

Este archivo contiene la clase message_model() y sus métodos que operan para obtener la información en la base de datos de la tabla mensajes o tareas. Está formada por:

- Propiedades:

- \$table: nombre de la tabla de la base de datos..
- \$connection: conexión a la base de datos.

- Métodos:

- getConnection(): instancia al objeto Connection e inicializa la propiedad \$connection.
- addMessage(\$message, \$nameEmployee, \$idUser): método para introducir un nuevo registro en la tabla mensajes.

- `getMessages()`: devuelve todos los registros de la tabla mensajes.
 - `completeMessageById($id)`: método que cambia el campo estado de la tarea o mensaje a completado en la base de datos.
 - `deleteMessageById($id)`: método que recibe el id del mensaje y lo elimina en la base de datos.
 - `getFoto($id, $name)`: método que recibe una id-user y nombre empleado y devuelve la foto de su tabla.
 - `functionTareasName($name)`: método que recibe un nombre de empleado y devuelve sus tareas o mensajes.
- **Carpeta operability:** en este directorio se encuentran todos los ficheros javascript, son los encargados de añadir nuevas funcionalidades o solucionar requisitos puntuales del funcionamiento de la aplicación. Es el añadido al modelo MVC.

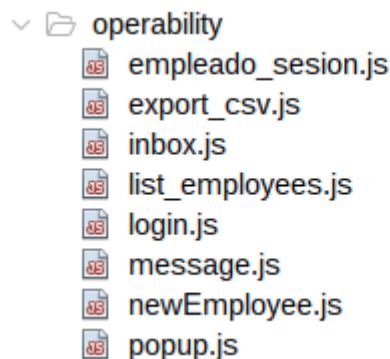


Figura 15. Vista del contenido carpeta operability

→ empleado_session.js:

Código js para listar las tareas según el trabajador seleccionado. Para el usuario empleado, en la vista general, al seleccionar el nombre del empleado se muestran sus tareas. Se realiza una petición por medio de `$.ajax()`, y se manda como parámetro el nombre del empleado seleccionado al método `listarTareasEmpleado` del controlador `message_controller`. Devuelve las tareas de este empleado.

→ export_csv.js:

Script javascript que se activa al pulsar un botón para exportar la tabla de tareas de los empleados a un formato legible por excel.

→ inbox.js:

En este archivo se añaden las funcionalidades a la vista inbox_employee.php (tabla con listado de tareas y datos de empleado).

- Checkbox de tarea completada que tacha el campo de mensaje o tarea y por medio de \$.ajax() realiza una petición asíncrona al método completeMessage de controlador message_controller para eliminarla de la base de datos.
- Filtro de búsqueda: método que al pulsar en botón filtrar muestra solo las tareas o mensajes del empleado seleccionado.
- Historial: método que hace visible en el dom la tabla con todas las tareas completadas.

→ list_employees.js:

En este archivo se definen dos funcionalidades, la primera, si se produce un click en el icono de tareas pendientes, te lleva a la vista de tareas, la segunda, al pulsar en un elemento con la clase borrar se realiza una petición por medio de \$.ajax() al método deletejs del controlador employee_controller, y este elimina al empleado con la id recibida como parámetro.

→ login.js:

En este archivo javascript se controlan las siguientes acciones que se producen en la vista login:

- Pulsar en el botón acceder: se comprueba con \$.ajax() el método checkAcces del controlador login_controller. Si los campos introducidos son los de la base de datos se da acceso a la aplicación web, si no, se recarga automáticamente la vista de login.

- Pulsar en nuevo registro: aparece un nuevo formulario de registro de usuarios.
- Pulsar en el botón registrar: se llama por medio de \$.ajax() al método addRegister, que añade un nuevo registro a la tabla user de la base de datos, del controlador login_controller.
- Pulsar en cerrar sesión: si pulsas en el salir de la sesión se ejecuta el método logOff que elimina la sesión del usuario y devuelve a la página de index.

→ message.js:

En este fichero javascript se añaden las funcionalidades al menú de iconos de la tarjeta de empleados. Se controlan los siguientes eventos:

- Pulsar en el icono de mensaje o tarea: muestra un text area donde escribir la tarea o mensaje.
- Pulsar en el icono de persona: despliega un popup con los datos generales del empleado, incluyendo su foto.
- Pulsar en enviar tarea: se manda por medio de \$.ajax() los parámetros id del empleado, nombre empleado y tarea encargada al método saveMessage del controlador message_controller.

→ newEmployee.js:

En este archivo se describe el código javascript necesario para validar la creación de un nuevo empleado en la base de datos:

- Primero se comprueba que el nombre no esté registrado en la base de datos, una vez pasada esta validación se habilita el siguiente campo.
- Para los campos puesto, email y subir foto se crean métodos específicos de validación. No podrán estar en blanco y el email debe seguir un patrón definido en una expresión regular.
- Si todas las validaciones son correctas, se habilita el botón de guardar, que crea un nuevo empleado.

→ popup.js:

En el fichero se encuentra el código javascript encargado de que al pulsar en el icono de ver la ficha o tarjeta de empleado se muestre toda la información específica de este mismo empleado en ella.

- **Carpeta resources:** se encuentran todos los recursos como imágenes de fondo o logos.
- **Carpeta styles:** esta carpeta contiene el archivo style.css (hoja de estilo) que define los estilos personalizados de la aplicación web.
- **Carpeta view:** en este directorio se encuentran las diferentes vistas o los archivos encargados de generar la interfaz de usuario.

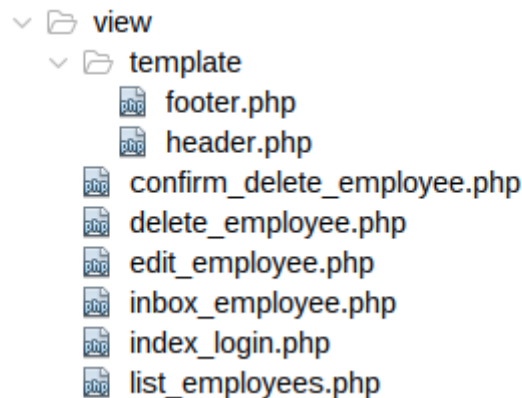


Figura 16. Vista del contenido carpeta view

- footer.php: este archivo se incluye en todas las vistas de la app y se encarga de cerrar las etiquetas body y html, además del div que tiene la clase container y que abrimos en el header. Se muestran logo y credenciales.
- header.php: aquí se abren las etiquetas necesarias de html y se añaden las llamadas a las librerías que vamos a utilizar. Es un

archivo común para toda la app y con él se muestra el menú de navegación.

- confirm_delete_employee.php: esta vista se mostrará cuando el usuario haga clic en eliminar un empleado.
- delete_employee.php: se ejecuta al eliminar un empleado. Nos muestra un simple mensaje informativo de que hemos eliminado el empleado y un link para volver al listado de empleados.
- edit_employee.php: este archivo muestra un formulario en el que podemos tanto crear un empleado, como editar uno ya existente.
- inbox_employee.php: vista que contiene dos tablas html, la primera se crea para mostrar las tareas o mensajes de los empleados y la segunda muestra el historial de tareas ya completadas. En esta vista se crean los siguientes métodos específicos:
 - `getTableMessage()`: método que instancia un objeto `message_ model`, llama al método que guarda el contenido de la tabla mensajes y lo devuelve en un array.
 - `listarTablaHistorial($tablaMessages)`: recibe la tabla mensajes de la base de datos y lista solo las tareas o mensajes completados.
 - `namesEmployeesTareas()`: método que muestra en el option del select los nombres de los empleados con tareas pendientes.
- index_login.php: vista que carga un documento html completo nuevo que contiene los formularios de autenticación y registro.
- list_employees.php: es la vista que se encarga de mostrarnos los empleados ya creados, si los hay. Utiliza el método `namesEmployeesTareas()` para mostrar en el option del select los nombres de los empleados.

2.2.2 Lenguajes de programación, tecnologías utilizadas y librerías externas:

- **PHP:** es un lenguaje de programación interpretado del lado del servidor y de uso general que se adapta especialmente al desarrollo web. Versión: PHP 8.2.6 (cli) (built: May 12 2023 06:23:43) (NTS).
- **JavaScript:** es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
- **HTML:** siglas en inglés de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web. Versión: HTML5.
- **CSS:** en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Versión CSS3.
- **Bootstrap:** es un framework de código abierto que utiliza diseños basados en CSS y HTML. Versión: 5.1.3.
- **jQuery:** es una biblioteca que facilita y agiliza la creación de páginas y aplicaciones web con JavaScript. Versión 3.6.3.
- **Font Awesome:** es un conjunto de herramientas de fuentes e íconos basado en CSS y Less. Versión: 5.8.1.
- **SQL:** es un lenguaje de consulta estructurado de dominio específico, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
- **MySQL:** es un sistema de gestión de base de datos relacional, multihilo y multiusuario.
- **PhpMyAdmin:** es una herramienta escrita en PHP con la finalidad de manejar la administración de MySQL a través de páginas web, utilizando Internet. Puede crear y eliminar bases de datos, crear, eliminar y modificar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia.

2.2.3 Base de datos

- Diagrama Entidad-Relación:

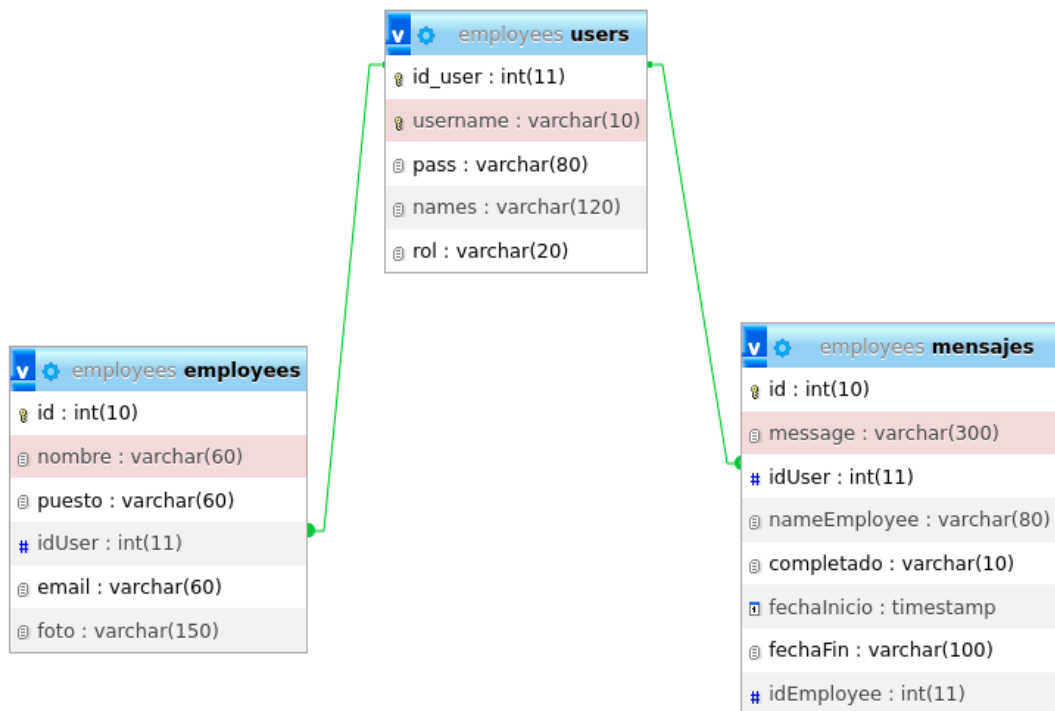


Figura 17. Esquema Entidad-Relación

- Nombre base de datos: employees.
- Tablas:
 - employees (id (clave primaria), nombre (campo único), puesto, idUser (clave foránea), email y foto).
 - mensajes (id (clave primaria), message, idUser (clave foránea), nameEmployee, completado, fechaInicio, fechaFin e idEmployee).
 - users (id_user (clave primaria), username (índice), pass, names y rol).

- Relaciones: la tabla users se relaciona con las tablas granada y mensajes con una clave primaria id_user, siendo clave foránea en estas dos.

2.3 Diseño

2.3.1 Diseño Responsive o adaptativo

Esta aplicación se basa en una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visitarlas. La vista es muy similar en ordenadores de escritorio, portátiles, tablets y móviles. Con este diseño se consigue redimensionar y colocar los elementos de la web de forma que se adapten al ancho de cada dispositivo permitiendo una correcta visualización y una mejor experiencia de usuario. Se caracteriza porque los layouts (contenidos) e imágenes son fluidos y se usa código media-queries de CSS3.

El diseño responsive permite reducir el tiempo de desarrollo, evita los contenidos duplicados, y aumenta la viralidad de los contenidos ya que permite compartirlos de una forma mucho más rápida y natural.

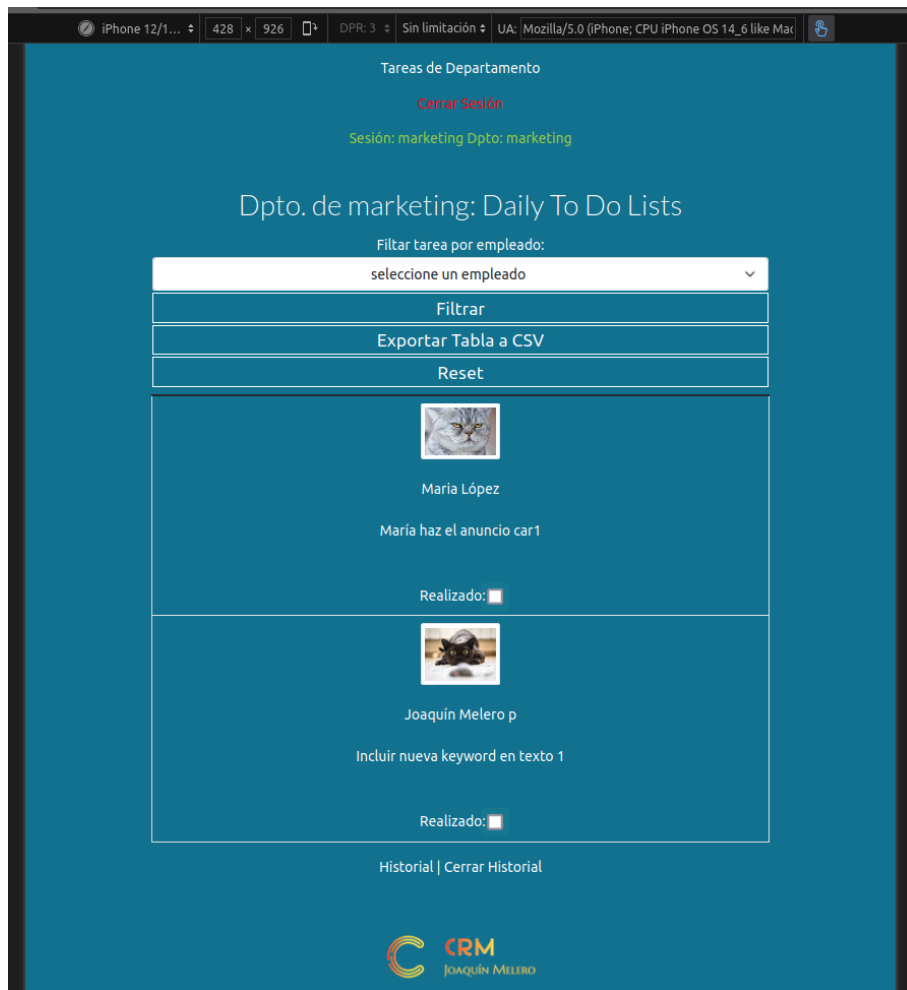


Figura 18. Vista simulada de iphone 12

2.3.2 Estilos

En este apartado se nombran los estilos generales utilizados en la app web:

- Tipo de fuente: Lato, importada de Google Fonts.
- Color de fuente principal: white.
- Tamaño de fuente: 90%.
- Color de fondo de la app: background: rgb (28, 114, 147).
- Botones: background-color: transparent y hover: #54dca4.
- @media screen and (max-width: 1400px)

3. Conclusión.

A continuación se expone una reflexión personal del trabajo realizado exponiendo dificultades y resultados.

3.1 Reflexión personal

Este trabajo es el fin a dos años de formación en el Ciclo de Grado Superior de Desarrollo de Aplicaciones Web. Con este proyecto se quiere reflejar gran parte de los conocimientos adquiridos durante este periodo y llevarlos a la práctica en una aplicación web diseñada desde cero con el objetivo de desarrollar una idea propia, incorporando conceptos como el de base de datos, programación orientada a objetos, diseño de interfaces web, validación de formularios o creación de login con autenticación.

La idea principal surge de la necesidad de gestionar un grupo de colaboración o de agilizar la comunicación entre empleados y coordinador/gestor. He intentado mantener siempre una visión profesional del programador web, utilizando técnicas modulares de estructuración para hacer un proyecto más legible, a la vez que seguro, y ganar así facilidad a la hora de interpretar y aportar mejoras al código por futuros programadores ajenos al proyecto.

Otro criterio básico que he querido seguir desde el comienzo de la elaboración, ha sido el de distribuir la ejecución del código de la aplicación equitativamente, entre el cliente y el servidor, esto es posible al utilizar los lenguajes de programación php y js. Me ha parecido que es una forma de conseguir una aplicación más eficiente y una forma de aprovechar tanto las prestaciones del servidor como el del usuario.

Tanto el diseño como las funcionalidades han sido elegidas personalmente, basándome en mi idea de lo que me parece más correcto o adecuado para este tipo de aplicaciones.

La mayor dificultad ha sido construir y estructurar la aplicación desde cero. Al no tener una referencia o framework de base, he tenido que crear directorios para ubicar los archivos siguiendo la lógica modelo-vista-controlador y, poder crear así, unas relaciones y llamadas a métodos lo más coherente posibles.

Otra dificultad importante es, que al ser un proyecto extenso, que se ha realizado en un tiempo determinado, he notado una cierta evolución en la forma de programar, por lo que lo correcto, una vez finalizada la app sería repasar y optimizar algunas líneas iniciales.

Mi conclusión final es que creo que el objetivo principal de este proyecto se ha cumplido satisfactoriamente y se ha conseguido crear una aplicación web funcional que combina varios módulos del ciclo. Sin embargo, el resultado puede parecer un poco básico, a nivel de opciones para el usuario que utilizará la app web, por lo que se añade un apartado de posibles vías futuras para seguir con su desarrollo.

3.2 Aprendizajes adquiridos

- Planteamiento, diseño y estructuración de una aplicación web.
- Trabajar y conectar archivos con diferentes códigos de programación.
- Incorporar librerías externas.
- Realizar llamadas de Ajax de modo asíncrono (async) para poder tener varias peticiones al servidor de manera paralela.
- Creación de login y registro de nuevos usuarios.
- Encriptación de contraseña en base de datos.
- Validación de formularios.
- Controlar eventos o sucesos específicos que ocurren en Javascript y asociarlos a una lógica de programación.
- Utilizar sesiones en php para almacenar datos de usuarios de manera individual usando un ID de sesión único.
- Definición de clases y métodos.
- Crear y aplicar estilos css propios a una interfaz de usuario.

3.3 Vías futuras de desarrollo.

Este apartado es para constatar una posible continuación o mejora de la aplicación web desarrollada en el proyecto en un futuro.

- El primer paso sería optimizar métodos, seguir modularizado procesos y limpiar el código duplicado.
- Mejorar el diseño para móviles y crear nuevas funcionalidades específicas para este tipo de dispositivos.
- Incorporar nuevas funciones en el acceso usuario, como incluir la posibilidad de comunicarse con el jefe de departamento o coordinador.
- Posibilidad de mandar un duplicado de las tareas o mensajes directamente por email.
- Incorporar un chat grupal en tiempo real.

4. Bibliografía.

- <https://www.php.net/docs.php>
- ECMAScript 6
- <https://www.w3schools.com/>
- <https://stackoverflow.com/>
- Temario 2022/2023 Ciclo Desarrollo de Aplicaciones Web
- <https://getbootstrap.com/>
- <https://developers.google.com/>
- <https://fontawesome.com/>
- <https://fonts.google.com/>
- <https://openwebinars.net/>

Joaquín Melero Peralta
jmelero76@ieszaidinvergeles.com