UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Aplicación web para la gestión del mantenimiento de vehículos

MEMORIA PRESENTADA POR:

David Molina Llácer

TUTOR:

Pau Micó Tormos

GRADO DE INFORMÁTICA

Convocatoria de defensa: Septiembre de 2018

Aplicación web para la gestión del mantenimiento de vehículos

David Molina Llácer

Resumen

Cuando llevas el coche al mecánico para hacerle el preceptivo mantenimiento el mecánico te lo devuelve con un cartón donde se apuntan los ítems revisados y cuando se ha de pasar la próxima revisión. También puede hacer estos apuntes en la factura. Normalmente ese cartón o factura se pierden y llevar el control del mantenimiento del vehículo se vuelve un infierno. En este TFG se propone el desarrollo de una herramienta web a partir de la cual el mecánico pueda gestionar los datos del mantenimiento de los vehículos de sus clientes de una manera fácil y eficiente. La aplicación ha de ser fácilmente configurable, con unas vistas de usuario sencillas y directas. Alternativamente los apuntes sobre el mantenimiento también se habrían de poder hacer desde un dispositivo móvil. La aplicación enviará un aviso automático al correo electrónico del cliente con una cierta antelación indicando la fecha de la próxima revisión. Sería una manera de fidelizar los clientes. De una manera secundária, esta tarea puede estar también integrada con la tarea de facturación.

Palabras clave: aplicación, mantenimiento, vehículos, web, mecánico.

Summary

When you take the car to the mechanic to do the required maintenance the mechanic returns it to you with a cardboard where the revised items are indicated and when the next revision is to be made. You can also make these notes on the invoice. Normally that cardboard or invoice is lost and to take control of the maintenance of the vehicle becomes a hell. In this end-of-degree project we propose the development of a web tool from which the mechanic can manage the maintenance data of their customers' vehicles in an easy and efficient way. The application must be easily configurable, with simple and direct user views. Alternatively, notes on maintenance could also be made from a mobile device. The application will send an automatic notice to the client's email with a certain advance indicating the date of the next revision. It would be a way to build customer loyalty. In a secondary way, this task can also be integrated with the billing task.

Keywords: application, maintenance, vehicles, web, mechanic.

Dirección/Supervision

Pau Micó

Contenido

1 Introducción

1.1 Antecedentes

Tras ver el trabajo que realiza constantemente un mecánico en su taller, observamos que más allá de su labor de mantenimiento de vehículos también realiza tareas de gestión las cuales le permiten llevar un control sobre el cliente. Es muy importante que el taller mecánico disponga de un sistema a través del cual el mecánico pueda llevar un seguimiento de todas las revisiones y reparaciones así como el acceso ágil a la información de determinado cliente.

Normalmente el mecánico recoge la información del cliente escribiendo en un formulario de papel, también es el caso de las revisiones que realiza diariamente. Este sistema que utiliza presenta varios problemas:

- Es posible que con el tiempo y debido al uso esas fichas de los clientes o de revisiones se terminan deteriorando o incluso se extravíen en el transcurso de su actividad.
- El acceso a esa información es un tanto lento ya que el mecánico ha de acceder al archivador y buscar entre todas las fichas hasta dar con el cliente y su historial.
- Toda esa información crecerá a medida que se realizan más revisiones o lleguen más clientes nuevos al taller, ello conlleva a que sea cada vez más difícil buscar algún registro y el volumen de papel cada vez sea mayor.
- Únicamente una persona puede hacer uso de la ficha de un cliente, de forma que cualquier actividad conjunta en el taller se volverá más pesada ya que los mecánicos tendrán que pasarse la ficha de la revisión de un vehículo cada vez que quieran realizar alguna tarea, ello conlleva que todavía sea más fácil que la ficha se extravíe.

Como solución se propone informatizar el sistema empleado a través de una aplicación web. El acceso por web permitirá acceder la información de cada cliente desde cualquier dispositivo y desde cualquier lugar. La aplicación también permitirá tener centralizada toda la información por lo que varias personas podrán acceder a los mismos registros con la información actualizada y trabajar conjuntamente. En cuanto al problema de espacio quedará más que resuelto ya que se podrán almacenar grandes cantidades de registros o sobre todo podrán buscar y encontrar rápidamente la información que sea necesaria.

1.2 Objetivos

El objetivo es crear una aplicación web para un taller mecánico la cual sirva como herramienta para la gestión sus clientes así como las tareas llevadas a cabo.

Esta aplicación permitirá por una parte manipular la información referente a cada cliente así como al registro de actividades relacionadas con él de forma que rápidamente podremos realizar una búsqueda mediante distintos filtros y entre diferentes módulos, es decir, no será necesario acceder siempre a la información de una actividad o de un vehículo buscando a través del cliente primero. Ahora podremos buscar a través de la matrícula de cualquiera de los vehículos de un cliente o por la última acción realizada sobre el vehículo de un cliente. De esta forma se expande las maneras de buscar un registro y al final ofrece mayor efectividad.

Una vez introducida la información del cliente se procede a la revisión del vehículo la cual podrá realizar cómodamente desde una tablet de manera que pueda tomar nota a medida que revisa el vehículo una vez almacenada la información cómodamente puede generar una documento PDF para imprimirlo o bien enviarlo por correo electrónico al cliente.

Una vez hecha la revisión se procede a realizar las acciones pertinentes sobre el vehículo tales como reparación, cambio de aceite, etc. En este punto el mecánico registrará todo lo que ha realizado y si hay algún comentario al respecto.

Finalmente el mecánico puede programar recordatorios de forma que al cliente se le notifique después un tiempo determinado que por ejemplo tiene que hacer el cambio de aceite o la próxima revisión.

En todo el proceso se pretende que haya comunicación total entre ambas partes de forma que el cliente tendrá acceso más restringido a la aplicación para que pueda consultar sus datos, revisiones realizadas, así como todas las acciones que se realizan a su vehículo e incluso pueda concertar una cita con el mecánico para una próxima revisión o reparación.

1.3 Requerimientos

Para llevar a cabo la programación de la aplicación voy a hacer uso del lenguaje de programación PHP para desarrollar la parte servidor la cual se encargará de manipular la información tanto de entrada como de salida hacia el navegador.

Para la representación de la información haré uso de lenguaje HTML, CSS y Javascript. Este último me permitirá automatizar tareas en la vista del navegador y mediante AJAX me permitirá establecer comunicación con el servidor.

2 Anteproyecto

2.1 Estado del arte

Tras un búsqueda por la red para ver aplicaciones de gestión de talleres mecánicos que pudieran haber programado antes he encontrado algunas varias que expongo a continuación:

1. **Tallermatic** ⇒ Es una *aplicación de escritorio* la cual permite el control de talleres de Mecánica en General, Chapa y Pintura, Electricidad, Neumáticos, Talleres de Motos, etc… Puede ser instalado en varios equipos y permite la sincronización de datos entre ellos mediante Dropbox, Google Drive o Onedrive.

2. **TallerGP** ⇒ Taller Gestión Profesional es una *aplicación web* para la gestión de talleres mecánicos online, con el que manejar todas las tareas administrativas de un taller. Con esta aplicación se puede gestionar las citas previas, visualizar la agenda, emitir presupuestos, albaranes, facturas, ordenes de reparación, enviar campañas de marketing por SMS o correos electrónicos, enviar facturas o presupuestos a los clientes, llevar un control del stock, tener un control documental de sus documentos importantes, como albaranes de compra, fichas de vehículos.

3. **GesTalleres** ⇒ GesTalleres es una *aplicación de escritorio* de Gestión y Administración de Talleres Mecánicos, con el que poder agilizar las labores administrativas relacionadas con el proceso documental de las reparaciones de vehículos.

2.2 Estudio de propuestas

2.2.1 Propuesta 1

Como primera propuesta se propone hacer uso del lenguaje de programación Javascript para realizar el back-end de la aplicación sobre un entorno alojado en un servidor propio de Node.js a través del cual habilitar un servidor HTTP.

Para desarrollar el front-end se propone el uso de lenguaje HTML y CSS para realizar el diseño de las páginas, acompañado de Javascript en este caso para realizar automatizaciones en la página así como la comunicación asíncrona entre el navegador y el servidor mediante AJAX.

Para el almacenamiento de la información se plantea utilizar MongoDB el cual es un sistema de Base de Datos NoSQL orientado a documentos.

2.2.2 Propuesta 2

Se plantea la posibilidad de hacer desarrollar el back-end mediante JSP el cual hace uso de Java como lenguaje de programación.

En el caso del front-end igualmente se propone HTML y CSS como lenguaje para desarrollar las páginas web y Javascript tanto para realizar automatizaciones como para realizar peticiones AJAX.

Esta vez, para el almacenamiento de la información se plantea utilizar MySQL como sistema gestor de Base de Datos haciendo uso del motor MyISAM.

2.2.3 Propuesta 3

Finalmente se propone hacer uso de PHP como lenguaje para desarrollar el back-end el cual irá acompañado de un servidor Apache.

Una vez más se propone un front-end programado en HTML y CSS acompañado de Javascript.

Para el almacenamiento de la información se plantea de nuevo utilizar MySQL como sistema gestor de Base de Datos haciendo uso del motor MyISAM.

2.3 Justificación

2.3.1 Estimación de recursos

Los lenguajes mencionados anteriormente tienen grandes comunidades de desarrollo detrás por lo que ofrecen muchas librerías las cuales nos permiten ahorrar tiempo además de abstraernos de temas más complejos para centrarnos en la aplicación en sí.

Para alojar la aplicación web se necesita un servidor web con diferentes características. En el caso de la primera opción necesitaremos un servidor el cual disponga de Node.js para realizar el servidor web y MongoBD para alojar la información. Si elegimos la segunda opción entonces será necesario disponer de un servidor con Apache Tomcat el cual interprete las páginas desarrolladas en Java y MySQL para almacenar toda la información. Finalmente si se utiliza PHP se requiere de un servidor con Apache y el intérprete de PHP así como de MySQL nuevamente para guardar la información.

2.3.2 Impacto económico

Todas las propuestas descritas anteriormente implican uso de aplicaciones open source por lo que en este caso se evitará pagar licencias de algún tipo.

A la hora de alojar la aplicación en un servidor se presentan diferentes precios. Por un lado podemos alojar la página en un servidor local como se ha visto en la primera propuesta pero surgen 3 problemas:

- En primer lugar es necesario disponer del propio servidor que en el caso de no disponer de uno se deberá adquirir.
- En segundo lugar se necesita de un administrador de sistemas para que realice el mantenimiento continuo.
- Finalmente pagar un pequeño extra para mantener la IP pública fija de manera que la dirección en la cual está alojada la página nunca cambie.

En el caso de la segunda y tercera propuesta por una cantidad mucho menor al mes podremos alojar la aplicación en un servidor remoto en el cual el mantenimiento nos lo gestiona dicha compañía tal como: Amazon Web Services, DigitalOcean, …

Quizá la más barata termine siendo PHP debido a que podemos encontrarlo preinstalado en la gran mayoría de hostings básicos que ofrecen las empresas de alojamiento web.

2.3.3 Propuesta final

Finalmente, me decanto por la tercera opción que es PHP dado que económicamente es la opción más barata y no por ello es que sea la peor, sino que es un lenguaje más extendido y la gran mayoría de compañías apuestan por este lenguaje.

Además, PHP tiene una larga trayectoria lo que ha generado una gran comunidad de desarrolladores la cual ofrece muchísimas librerías gratuitas así como gran cantidad de ayuda por los foros. Entre una de las extensiones desarrolladas en PHP por la comunidad está la que nos permite conectar con MySQL de forma muy sencilla y sobre todo rápida.

3 Implementación

3.1 Entorno de desarrollo

Para desarrollar la aplicación voy a hacer uso de Atom. **Atom** es un editor de código de fuente abierta con soporte para plug-ins y desarrollado por GitHub. Con los complementos predeterminados es compatible con una gran cantidad de lenguajes de programación como: HTML, CSS, C/C++, C#, Java, Javascript, JSON, Python, PHP, Perl, XML, SQL, ... En este caso haremos únicamente uso de HTML, CSS, PHP, SQL y JSON para el intercambio de información entre cliente y servidor.

A la hora de realizar las pruebas me he montado un entorno local mediante Docker. **Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux. Aislando los recursos del kernel de Linux logra permitir que diferentes contenedores se ejecuten dentro de una sola instancia de Linux por lo que logra evitar la sobrecarga de iniciar y mantener máquinas virtuales.

Finalmente para llevar a cabo el seguimiento de cambios del proyecto hago uso de Git. **Git** es un software de control de versiones, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Mediante Git podré subir el código fuente a un repositorio online a través del cual compartir el código tanto con el tutor como con el resto de personas interesadas.

3.2 Implementación práctica

Para llevar a cabo el desarrollo de la aplicación como paso previo he desarrollado un pequeño **Framework** el cual me permitirá más adelante desarrollar y ampliar fácilmente módulos sobre la aplicación. En el desarrollo de software, un **Framework** es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

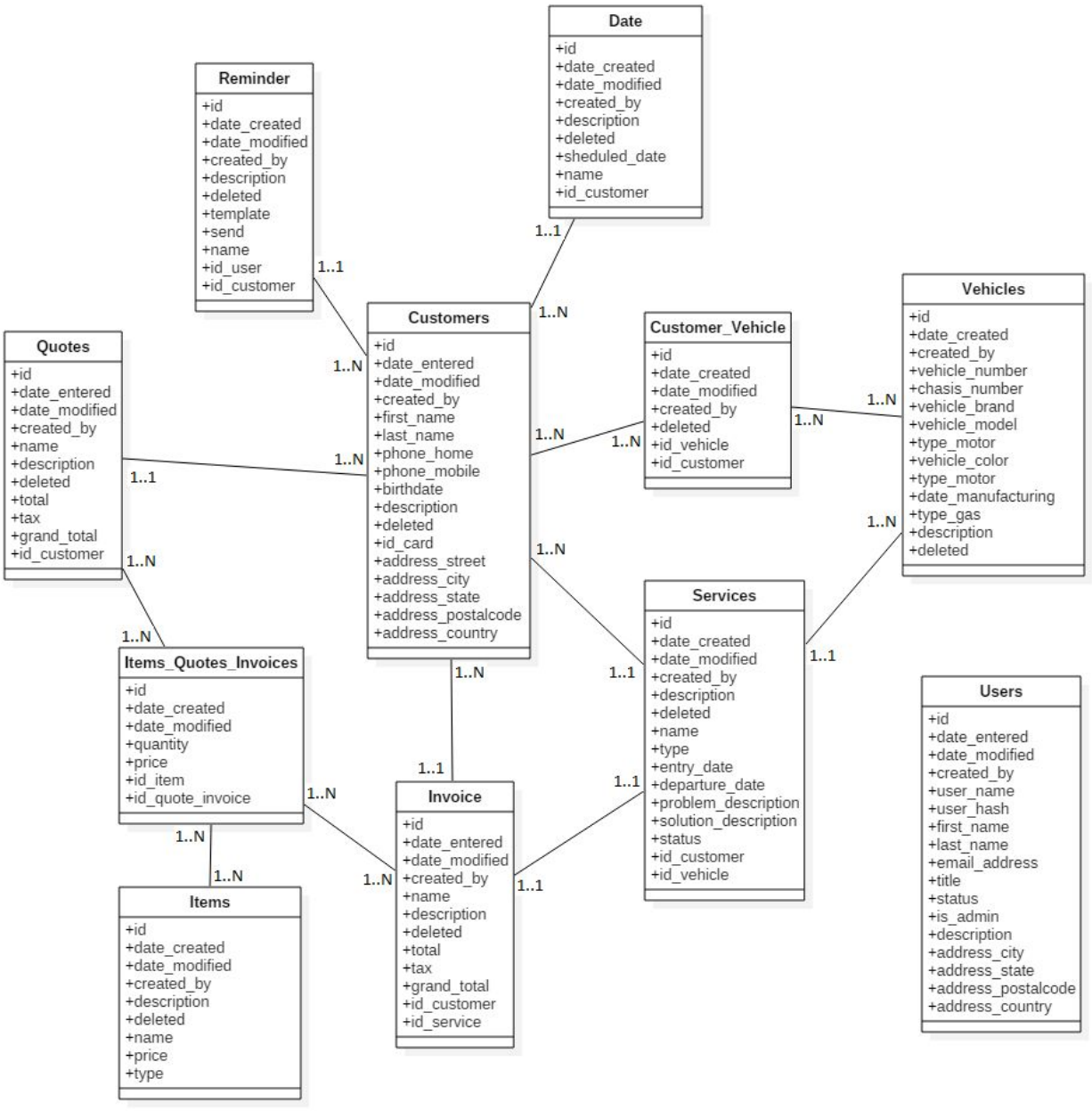
Este **Framework** ha sido desarrollado usando un patrón **Modelo–vista–controlador** (**MVC**). Este patrón separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello **MVC** propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

3.2.1 Diseño del producto

Una parte fundamental de la aplicación es la base de datos en la cual registramos toda la actividad. La base de datos nos permitirá ingresar datos de vehículos, clientes, citas, recordatorios, servicios, presupuestos, facturas e items. Nos ayudará en la ardua tarea del registro de los mismos de una manera mucho más eficiente y conveniente para la empresa.

Para esto he desarrollado el uso de tablas de referencia de cada una de las posibles entidades que se nos presente, además estas contendrán una serie de campos los cuales nos informarán detalladamente durante el transcurso de la actividad. Cada uno de los módulos de la aplicación representará una tabla.

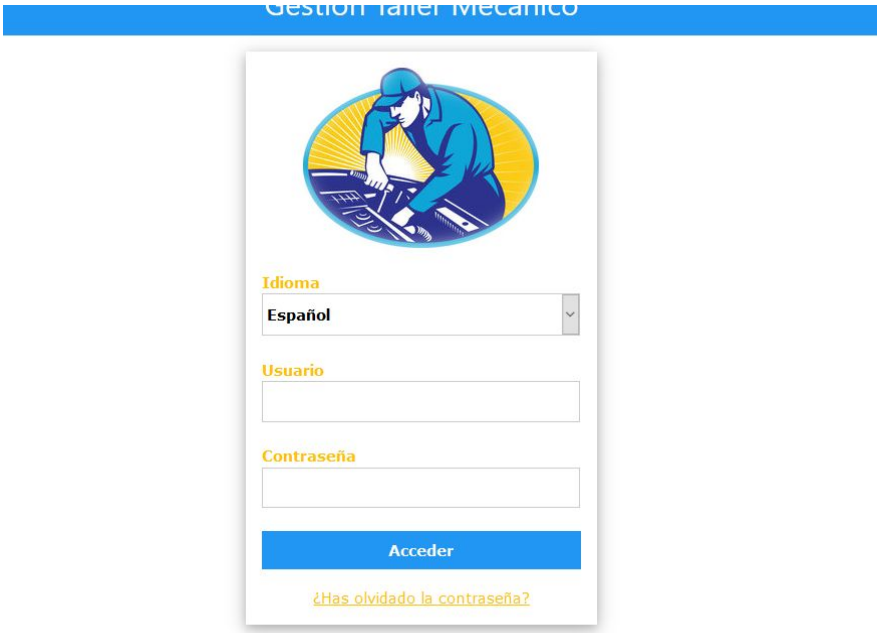
En la imagen que vemos a continuación podemos ver visualmente todas las relaciones que se presentan en la base de datos de este proyecto estas se presentan en 2 formas: 1..1 es que para la primera tabla tendrá una única asociación con la otra tabla. La otra es 1..N la cual representa que una tabla puede tener asociaciones con varias registros de la otra tabla.



Por último cabe destacar que a pesar de que no se vea relaciones en la tabla de usuarios no significa que no tenga. Usuarios principalmente nos permite consultar el usuario y la contraseña para acceder a la herramienta. No obstante tiene relación con todas las tablas ya que internamente cada uno de los registros que se crean en la base de datos lleva el id del usuario que ha creado el registro. En este caso lo he mostrado aislado por que realmente quiero que se vea bien todas las relaciones sobre la tabla de Clientes que es la principal.

3.2.2 Maquetas

En primer lugar la primera vista que nos aparece es la página de acceso en la cual ingresamos a través de un formulario básico el usuario y la contraseña.

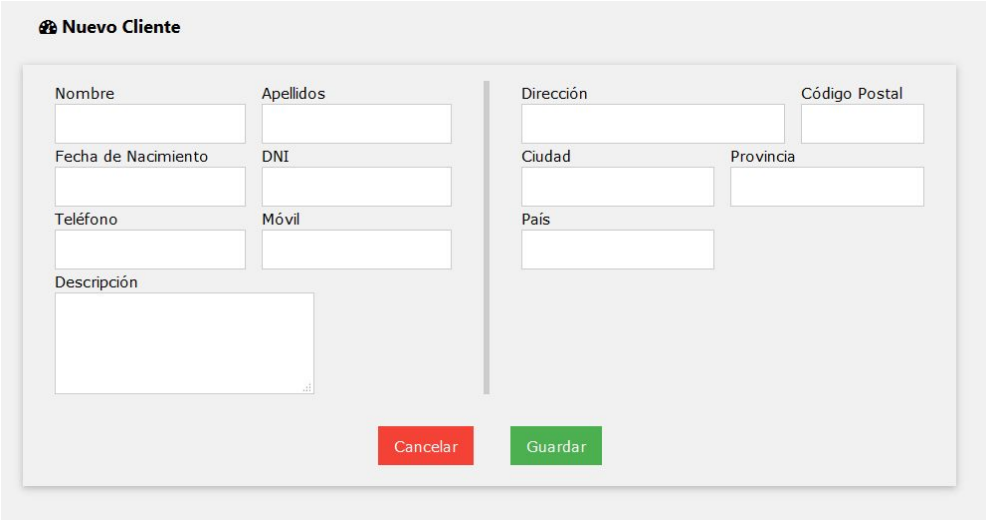


Una vez identificado correctamente observamos la vista principal a través de la cual nos moveremos continuamente por la aplicación. Esta página está comprendido por el menú lateral el cual nos acompañará durante todo tiempo que estemos usando la aplicación y una página principal la cual nos muestra un resumen de las actividades realizadas y por realizar.

Dado que la base de datos es un pilar fundamental en la aplicación y hemos de estar continuamente añadiendo, modificando y eliminando información he seguido el método CRUD (Crear, Leer, Actualizar y Borrar) por lo cual estas representarán las vistas principales de la aplicación. Todos los módulos se basarán en dos vistas ya que Crear y Actualizar los he unido en la misma vista y a través de la vista de lectura se pueden eliminar los registros. La diferencia entre cada módulo son los campos que contienen, cada uno contiene tipos de información distinta.

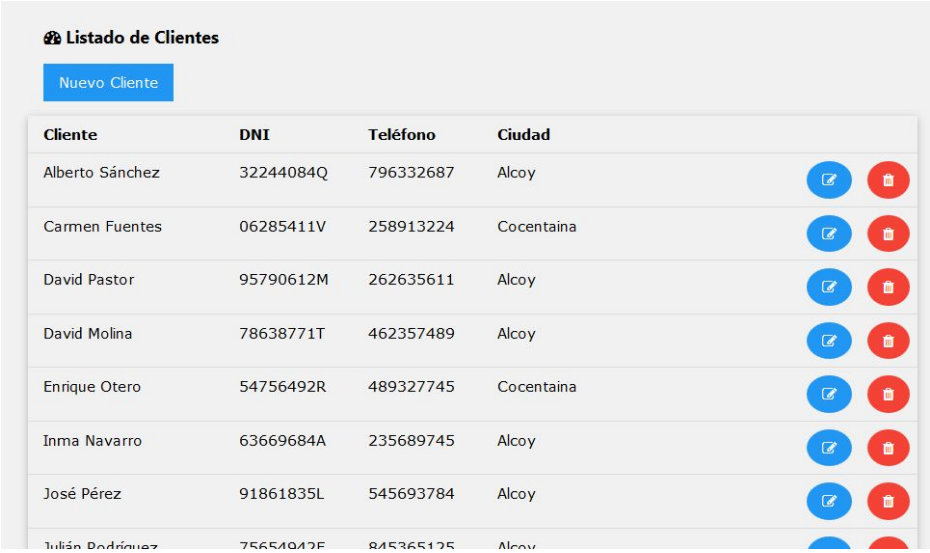
Vista de Creación y Actualización

Esta vista se basa en un formulario web en la cual se añade la información para ser almacenada en la Base de Datos. Si creamos un registro nuevo el formulario aparecerá totalmente en blanco para ser rellenado. En caso de estar modificando el diseño es completamente el mismo solo que los campos en los que previamente se hubiesen introducido datos ahora aparecen esos datos previos.



Vista de Lectura

La vista de Lectura está comprendida por dos vistas ya que en primer lugar se muestra un listado con todos los registros introducidos. También la vista de Lectura nos permite eliminar los registros seleccionados. La vista de Lista consiste en una tabla con la información básica más destacada representado en columnas y después en cada fila se sitúa cada uno de los registros. Desde la vista de lista podremos eliminar cada uno de los registros que se muestran en la tabla.

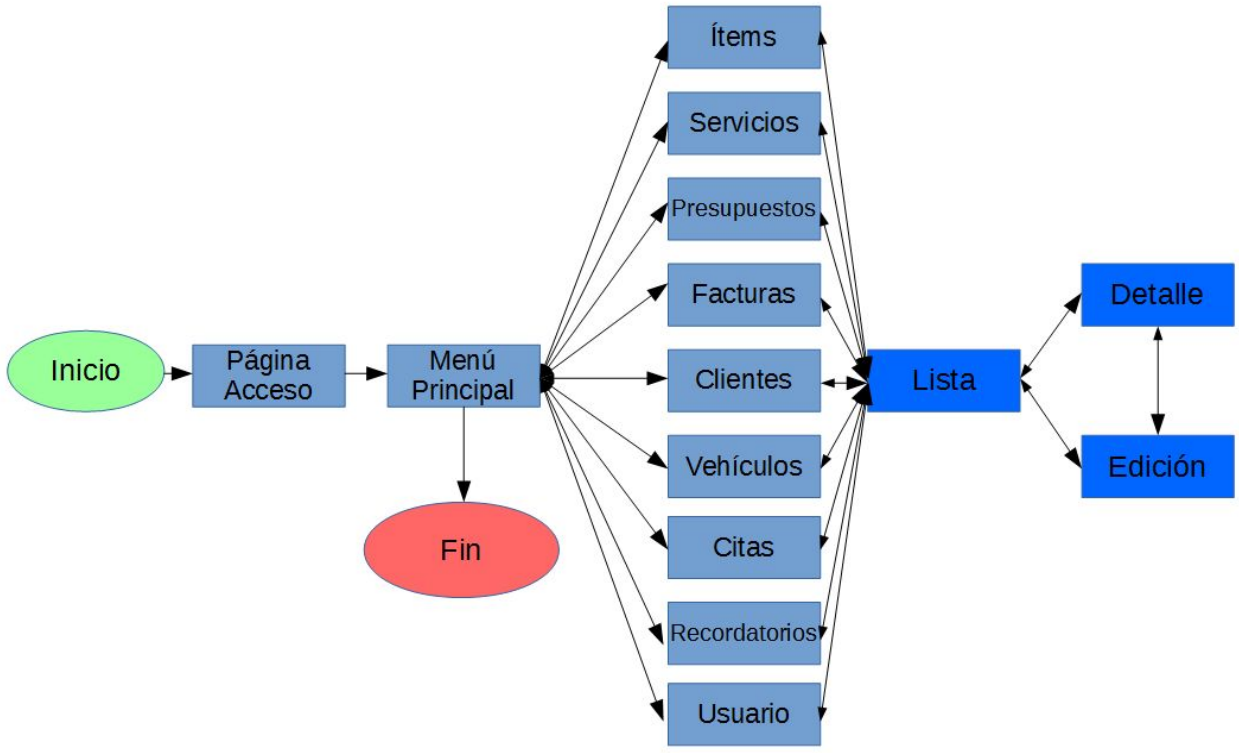


Una vez pinchamos sobre un registro accedemos a la vista de Detalle, dentro de cada registro podremos observar una página con los distintos atributos de ese registro. También podremos eliminar ese registro en sí desde esta vista.

3.2.3 Diagramas de funcionamiento

Como he mencionado anteriormente más allá de la vista de acceso y de la página principal se repiten pero siendo personalizados a cada módulo o tabla. Eso quiere decir que una vez accedemos a cada módulo las opciones que se nos presentan son: Crear un registro nuevo, Modificar el registro, Visualizar el registro y Eliminar el registro. También dado que en todo momento tenemos accesible el menú podremos acceder a todos los módulos desde la misma ventana, eso hace más cómodo navegar por la aplicación.

Este sistema de navegación hace que el funcionamiento de la herramienta parezca más estilo escritorio ya que en ningún momento se hace refresco de la pantalla. Las funciones del frontend se encargan de hacer las peticiones por detrás comunicándose con el backend y cuando obtiene la información la muestra por pantalla sin haber tenido que cargar de nuevo la página en el navegador.



3.3. Pruebas

Una vez obtenido todo un sistema completamente funcional, dado que se ha desarrollado sobre un ordenador local y se ha revisado sobre este mismo, ahora procedo a realizar una serie de pruebas las cuales permitirán detectar posibles errores en el sistema en el cual esta aplicación ejercerá su función.

La máquina donde se hará la instalación de la plataforma y donde quedará publicada es a través de un hosting compartido el cual ofrece: Una única CPU, 512 MB de memoria RAM, 25 GB de almacenamiento y una base de datos. Por el momento dado que es una aplicación de uso reducido a un taller será más que suficiente, no obstante a pesar de ello las siguientes pruebas me ayudarán a saber si la aplicación se adapta a la máquina y cumple con la función de gestionar un taller mecánico..

3.3.1 De sistema

En primer lugar vamos a ver hasta qué punto puede dar de si la máquina y para ello vamos a estresarla para ver cómo nos responde en esas situaciones. Como he mencionado antes, el uso estará limitado a un número determinado de usuarios que serán los empleados del taller. Aun así, testeamos la máquina saurandola con un número muy elevado de peticiones de forma que podemos ver si el servidor en primer lugar responde correctamente a las exigencias actuales y además cuánto margen hay para que en un momento dado el número de usuarios pueda crecer.

Para proceder a las pruebas de estrés y de carga vamos a hacer uso de la herramienta **Apache Benchmark**, un software libre distribuido bajo los términos de la Licencia Apache. A pesar de que lleve el nombre de Apache esta herramienta nos permite medir el rendimiento de cualquier servidor, no solo de Apache.

Lo primero de todo una vez descargado apache es localizar en la carpeta “bin” la aplicación, ésta está bajo el nombre **ab**. Una vez localizada la aplicación ya podremos hacer uso de ella a través del siguiente comando.

```
ab -c 10 -n 1000 http://tfg.davidmolina.net/
```

Con esta simple instrucción generamos 1000 (a través del parámetro -n) llamadas a la URL que especificamos al final de la instrucción. Estas llamadas serán distribuidas a través de 10 hilos (a través del parámetro -c), precisamente esta capacidad de concurrencia nos permitirá comprobar condiciones de carrera o bloqueos ya que el comportamiento de las peticiones es más natural que si se realizan las 1000 seguidas en un bucle.

Una vez lanzada la instrucción estos son los resultados obtenidos:

```
Server Software:        Apache
Server Hostname:        tfg.davidmolina.net
Server Port:            80

Concurrency Level:      10
Time taken for tests:   85.447 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      2112446 bytes
HTML transferred:       1657000 bytes
Requests per second:    11.70 [#/sec] (mean)
Time per request:       854.470 [ms] (mean)
Time per request:       85.447 [ms] (mean, across all concurrent requests)
Transfer rate:          24.14 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       47   83  99.1     78    3125
Processing:    78  766 350.4    703    4657
Waiting:       63  497 328.6    484    3719
Total:        172  849 367.4    781    4766

Percentage of the requests served within a certain time (ms)
  50%    781
  66%    813
  75%    859
  80%    891
  90%   1000
  95%   1219
  98%   1500
  99%   3704
 100%   4766 (longest request)
```

Tras la prueba, en primer lugar el dato más interesante a observar es el número de **peticiones por segund**o que es capaz de procesar, en este caso son **11** y el **tiempo medio de carga** de la página es de **781 milisegundos**.

Ahora ya somos capaces de conocer un poco más los límites de este servidor. En el peor de los casos, para que el funcionamiento sea óptimo el número de usuarios que pueden hacer uso de la aplicación son 11 más o menos por lo que si algún día aumentase el número de empleados habría que plantearse cambiar a un plan de hosting con mayores prestaciones.

3.3.2 De integración de sistemas

Una vez obtenidos los datos sobre la máquina que se aloja se procede a mover todo el contenido al servidor remoto en el cual se hará uso de la aplicación. Con un cliente de FTP tal como **FileZilla** subimos rápidamente los ficheros y mediante la herramienta **phpMyAdmin** subimos accedemos a la base de datos para acceder a la base de datos y volcar todo la estructura almacenada en un fichero SQL.

Nada más introducir la URL y acceder a la web se nos muestra la pantalla de identificación. Hasta aquí todo correcto, el servidor responde bien y el framework nos devuelve la página correcta. El error llega cuando se intenta acceder a la aplicación, está enseguida deja de funcionar y nos muestra una página en blanco. Previamente al acceder con el usuario la base de datos no nos ha lanzado un mensaje de error ya sea porque el usuario o la contraseña no estaba bien, o simplemente porque no ha podido acceder a la base de datos. Eso me hace pensar que si que se ha identificado correctamente y a accedido a la página principal por lo que reviso en seguida que puede estar pasando allí.

Tras un rato el error queda localizado. Debido a que el desarrollo ha sido llevado a cabo en un entorno local mientras se desarrollaba la aplicación, las rutas de los enlaces no corresponden ya que cambia la dirección web. Tras revisar ciertas páginas web para dar con la solución vi que PHP incluye unas cabeceras las cuales nos pueden proporcionar la información necesaria. Estas cabeceras son '**SERVER_NAME**' y '**PHP_SELF**'.

'**SERVER_NAME**' me devuelve el nombre del host del servidor donde se está ejecutando actualmente el script. Si el script se ejecuta en un host virtual se obtiene el valor del nombre definido para dicho host virtual.

'**PHP_SELF**' me muestra el nombre del archivo de script ejecutándose actualmente, relativa al directorio raíz de documentos del servidor.

Ambas en combinación me devuelven dinámicamente la URL que corresponde en cada uno de los servidores en los que se está ejecutando la web por lo que almacenando el valor en

una variable y añadiendo la extensión que corresponda dentro de cada enlace queda el problema resuelto. Además ahora podemos mover la aplicación a cualquier servidor de manera que seguirá funcionando de igual manera.

Ahora que ha quedado resuelto el problema del acceso podemos continuar revisando el servidor para ver si responde tal y como estaba previsto.

Navegación: Todos los enlaces son accesibles sin ningún problema. El servidor nos devuelve las páginas de cada uno de los módulos así como los de la página de administrador. En este punto gracias al problema anterior corregido no se ha vuelto a experimentar problema alguno al acceder a alguna dirección web existente.

Creación: Habiendo accedido a los diferentes módulos se prueba que se puedan crear registros nuevos, además gracias a las nuevos atributos de HTML5 comprobamos que además nos deja introducirlo en el formato correcto ya que sino, no nos deja subir el registro. También haciendo uso de las consultas SQL no hay ningún problema para subir los diferentes registros a la base de datos.

Listado: Tras acceder a las distintas páginas donde se muestran los registros vemos que la aplicación no tiene ningún problema para devolverme la información sirviéndose de la base de datos para ello.

Edición: habiendo creado los registros y luego habiendolos listado también comprobamos que podemos acceder al modo de edición y ya que a pesar de que es la misma página que la de creación, aquí se revisa que los campos de introducción de datos están rellenados con la información correcta del registro al que pertenece. En este punto se a puesto especial incapié al módulo de Presupuestos/Facturas ya que al editar interviene un código javascript que genera la estructura correctamente con los datos la tabla intermedia correspondiente.

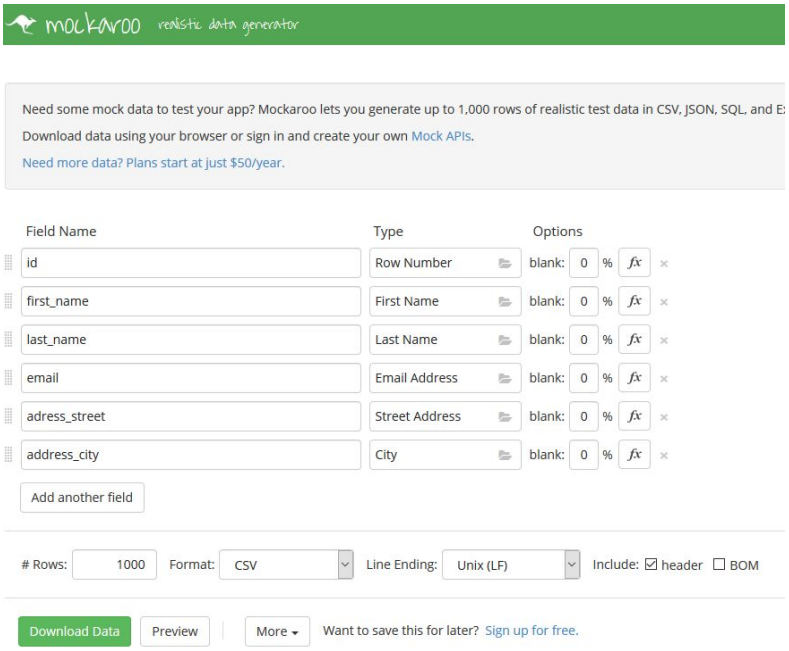
Detalle: Al acceder a la vista de detalle de cada módulo vemos que muestra todos los datos correctamente salvo un pequeño error. Las fechas están en el formato que usa la base de datos, en este caso el formato Americano. Aquí lo que esperamos es obtener el resultado en el formato de fecha de España y para ello gracias a las funciones de PHP de la librería de fechas este pequeño percance queda resuelto. Por lo demás vemos como el resto de la página se muestra correctamente, incluidos los subpaneles con las relaciones a los otros módulos así como los enlaces a ellos.

Dashlets: Algo también a revisar finalmente es que las pequeñas ventanas de la página de inicio accedan a la base de datos correctamente. Ya que de esta forma podemos averiguar si las diferentes estadísticas son correctas ya que de lo contrario toda esta información sería inútil. En efecto todos están funcionando correctamente por lo que muestran información en

tiempo real correctamente. Quizás una única cosa que comentar por nombrar algo es un cambio de etiqueta que no se mostraba correctamente, al cambiar el nombre de un módulo esta se quedo así, lo que provocaba confusión.

3.3.3 De volumen

Las pruebas de volumen hacen referencia a grandes cantidades de datos para determinar los límites en que se causa que el sistema falle. También identifican la carga máxima o volumen que el sistema puede manejar en un periodo dado. Por ejemplo, si el sistema está procesando un conjunto de registros de Base de datos para generar un registro, una prueba de volumen podría usar una Base de datos grande de prueba y verificar que el sistema se comporta como debe y crea el registro correctamente.



En primer lugar para ello voy a hacer uso de un generador de registros falsos. El nombre de la aplicación es Mockaroo y lo que hará es generar una larga lista de registros en JSON. En la aplicación simplemente añadimos las columnas que queremos generar y el tipo de valor que nos ha de rellenar. Después ya podremos descargar el listado de registros para su posterior subida a la base de datos.

Finalmente gracias a un script recogemos esa información y mediante un bucle vamos recogiendo cada línea del código para subirla mediante una consulta SQL a la base de datos. El proceso es gradual de forma que observamos cómo reacciona la base de datos.

Una vez terminada de subir toda la información a la base de datos lo siguiente es medir el tiempo que tardamos en obtener los resultados. Con ello mediremos el tiempo y además el tamaño que ocupa en la base de datos para ver si puede llegar al límite de espacio pronto.

Con la información recogida muestro a continuación un tabla donde se recoge todas las mediciones realizadas sobre la base de datos. En esta tabla se recoge la información promedia entre todas las tablas que componen la base de datos de la aplicación.

Nº Registros	Tiempo de Consulta	Espacio en Disco
10.000	0,016 segundos	4,0 MiB
100.000	0,047 segundos	43,2 MiB
200.000	2,719 segundos	80,3 MiB
300.000	3,984 segundos	117,4 MiB
400.000	5,156 segundos	155,6 MiB
500.000	6,546 segundos	194,8 MiB

Trás la medición podemos observar que hasta los 100.000 registros con el sistema que tenemos actualmente la aplicación se puede mover sin problemas. El problema aparece cuando pasamos a 200.000 que el tiempo de respuesta se dispara bastante. Llegado el caso en que la aplicación mueva tal volumen de registros se planteará un cambio de máquina capaz de mover más información en menor tiempo.

En cuanto al espacio en disco a pesar de que ocupa unos cuantos megas nunca supondrá un problema ya que en la máquina actual disponemos de 25 GB y la aplicación tan solo ocupa 35,8 MB, así que dificilmente llegará al limite de capacidad de disco.

4 Resultados

4.1 Migración al entorno de producción

(**PHP, MySQL, Apache**)

Para llevar a cabo la migración hay que proceder de la misma manera que cualquier aplicación web, hay que copiar el contenido de la carpeta mediante FTP o directamente si estuviésemos en la misma máquina la cual va a hacer de servidor de producción. De esta manera funcionará sin problema alguno pero se recomienda evitar copiar algunos ficheros que son utilizados durante el desarrollo de la aplicación.

Por ejemplo Git nos crea ficheros en los cuales gestiona las versiones que han habido hasta la fecha. Por ello siempre será mejor evitar copiar el directorio .git y el fichero .gitignore a través del cual evitamos que ciertos ficheros sean tenidos en cuenta a la hora de subir nuevo contenido al repositorio.

Además de Git, la propia aplicación genera ficheros que en el entorno de producción ya no nos hacen falta porque una vez se esté ejecutando la aplicación en el entorno de producción ya generará de nuevo esos ficheros. Por ejemplo, una de ellas es el directorio de caché, ésta se genera cada vez que un cliente pide una página de forma que al quedar registrada en caché si otro usuario a de acceder de nuevo la carga será más rápida ya que el servidor devolverá esa misma página sin tener que procesarla de nuevo.

Para estar al tanto de los eventos de la aplicación, durante su ejecución se vuelca todos los eventos en un fichero de log el cual si evitamos mover no arrastraremos toda la información del entorno de desarrollo que junto con la de producción resultará un estorbo.

Finalmente un fichero que también es recomendable evitar copiar es el de base de datos. Es un fichero con extensión “sql” la cual contiene la información de la base de datos en la cual se ejecuta la aplicación. Este fichero solo tiene un uso la primera vez que instalamos la aplicación en el servidor, una vez volcadas las tablas este fichero no nos cumple ninguna labor más en el directorio principal.

4.2 Manual de usuario

En este apartado revisaremos paso a paso cómo hacer uso de la aplicación, en primer lugar veremos cómo instalar la aplicación para que funcione correctamente y después paso a paso cómo podemos movernos por la aplicación en sí.

4.2.1 Instalación de la aplicación

Para realizar la instalación de la aplicación serán necesarios tres pasos a través de los cuales volcaremos todos los directorios, prepararemos la base de datos y finalmente configuraremos una tarea automatizada en el servidor la cual nos permitirá realizar los recordatorios.

El primer paso como se ha mencionado antes es volcar el directorio principal, para ello podemos hacer uno de un cliente FTP como: “FileZilla”, “Transmit”, “Cyberduck” o “SmartFTP”. Estos clientes FTP nos va a permitir transferir los ficheros al servidor. El más usado es “FileZilla” pero el resto es igualmente válido, incluido otros tantos que no están en la lista.

El segundo paso es preparar la base de datos sobre la que la aplicación realizará las consultas. En el directorio principal hay un fichero con extensión “sql” el cual contiene las instrucciones en lenguaje SQL el cual nos creará la base de datos así como las tablas referentes a cada módulo. Si tenemos instalado “phpmyadmin” podremos acceder y desde el panel podremos realizar la consulta para que se generé todo. En caso de no disponer de la herramienta web “phpmyadmin” podremos hacer uso de otras aplicaciones de escritorio sobre las que hacer las consultas tales como: “MySQL Workbench” o “HeidiSQL”.

Finalmente, ya tendremos preparada la aplicación web pudiendo acceder con cualquier navegador introduciendo la dirección web. El único problema ahora es que los recordatorios jamás funcionarán debido a que no se está comprobando en ningún momento cuando se deben de activar. Para ello activamos un proceso en segundo plano que ejecute un fichero continuamente. Esta tarea es distinta entre sistemas Unix y Windows así que a continuación detallaré cómo realizar cada una de ellas.

Entorno Unix

En el sistema operativo Unix las automatizaciones se realizan mediante un cron. Cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.

Para añadir la tarea al cron basta con ejecutar el siguiente comando:

```
crontab -e
```

Abrirá el editor definido en la variable de entorno EDITOR y cargará el fichero crontab correspondiente al usuario que está logueado. Una vez en el editor bastará con añadir la siguiente línea al final:

```
*/15 * * * *  php <RUTA HASTA APLICACIÓN>/cron.php
```

Una vez introducido el comando sustituyendo previamente <RUTA HASTA APLICACIÓN> por la ruta en la cual el servidor tiene la aplicación alojada. Esto hará que cada 15 minutos revise a través del fichero "cron.php" si hay algún recordatorio pendiente de avisar enviando los mails correspondientes.

Entorno Windows

En entornos Windows la manera en que se configura cambia debido a que se realiza a través de la interfaz. Aquí las automatizaciones se llaman "Tareas Programadas de Windows".

En primer lugar hay que acceder a la aplicación "Programador de Tareas" o "Task Scheduler" desde Inicio -> Programas –> Accesorios -> Herramientas del Sistema -> Tareas Programadas.

Una vez en la aplicación pulsamos sobre "Crear tarea" de forma que se nos desplegará una ventana nueva para introducir los datos. Desde la pestaña "General" le asignamos un nombre y una descripción a la tarea. Ahora en la pestaña de "Acciones" agregamos una y en la siguiente ventana le asignamos la acción en este caso "Iniciar un Programa" y el script a ejecutar en este caso el fichero "cron.php" mediante el intérprete de PHP.

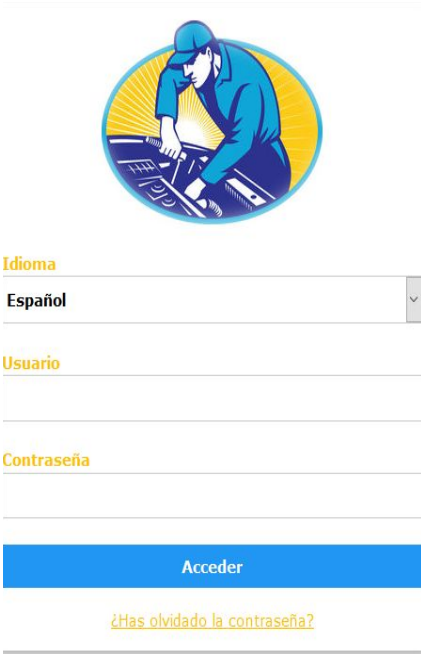
Para finalizar ya solo queda especificar cuándo se va a ejecutar esa tarea, desde la pestaña "Disparadores" estableceremos el tiempo. Lo configuraremos de forma que se ejecute diariamente cada 15 minutos.

4.2.2 Explotación

Acceso como Usuario Normal

Acceso a la Aplicación

En primer lugar la primera vez que ingresamos en la aplicación se nos presenta la pantalla de inicio de sesión. Esta página tal y como su nombre indica nos permitirá identificarnos para acceder a la aplicación. Lo primero de todo es seleccionar el idioma correcto en el desplegable superior de entre los que se presentan. Esto ya nos permitirá hacer uso de toda la aplicación con el idioma establecido. Para acceder hay que introducir los datos de acceso (Nombre y Contraseña) en el formulario y pulsar sobre el botón de acceder. Si es la primera que vez que accedes a la página de inicio de sesión probablemente no dispongas de un usuario y contraseña por defecto para acceder a la aplicación, en ese caso deberás introducir el usuario por defecto cuyo nombre es “admin” y su contraseña es “admin”. Recuerda que una vez accedido por primera vez a la aplicación deber de cambiar la contraseña por otra diferente para evitar accesos indebidos.



En caso de ingresar un usuario o una contraseña incorrecta el navegador mostrará una ventana emergente para informarte del error.

Página Principal

Una vez identificado correctamente el usuario, el sistema muestra la página principal la cual consta de dos partes:

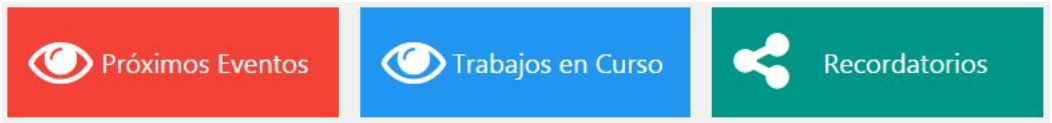
Actualmente, estamos en la página de Resumen que marca el menú. A la parte izquierda de la pantalla se nos muestra el menú principal, este nos acompañará continuamente de ahora en adelante y desde él podremos movernos a cualquier parte de la web. Existen ciertas restricciones ya que dependiendo del rol de tu usuario hay módulos que resultan inaccesibles, la aplicación no

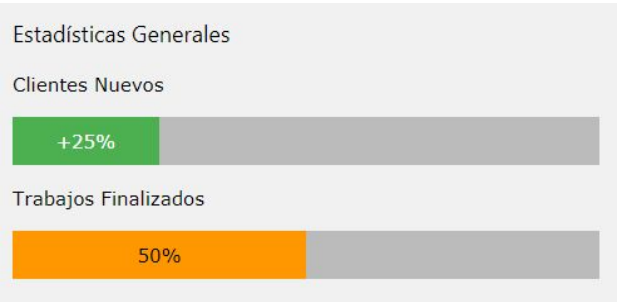
los mostrará, en este caso ocurre con el módulo de Ítems. En la parte superior se muestra una imagen y el nombre del usuario que hay registrado en este momento la aplicación. Desde la parte superior tendremos 3 accesos directos:

- El primero nos lleva a las citas y recordatorios que hay registrados por el momento en la aplicación.
- El segundo nos lleva a los dos módulos de administración: Usuarios y Items. Este acceso es exclusivo para los usuarios con rol de administrador por lo que si accede un usuario normal no se le mostrará esta opción.
- El tercero es el botón para cerrar la sesión. Si lo pulsamos nos llevará de nuevo a la ventana de acceso.

En la parte inferior tenemos un listado de los módulos disponibles a los que podemos acceder, si pinchamos en cualquiera de ellas nos llevará a la vista de lista del módulo seleccionado. Finalmente como este menú tiene un diseño adaptable, las opciones ocupan el ancho de la página. Además puede plegarse y desplegarse en pantallas con resoluciones más pequeñas, esto lo haremos pulsando sobre el botón superior con tres barras horizontales.

A la parte derecha de la página principal se nos mostrará la propia página en si, esta consiste en un resumen de las actividades registradas en la aplicación. Por ejemplo en la parte superior disponemos de una tabla que nos muestran lo eventos próximos, los eventos que están en curso y finalmente algún recordatorio que esté cercano. Si pinchamos sobre cualquiera de ellos nos llevará directamente al detalle de ese registro por si necesita ampliar la información.





A continuación tenemos una serie de estadísticas las cuales pueden servir de ayuda al usuario para conocer el estado actual de su taller en cuanto a clientes por ejemplo, cuantos han ingresado nuevos o

también cuánto trabajo ha hecho nuevo este mes.

Por último tenemos un listado con los últimos usuarios que han estado accediendo a la herramienta. Es sencillo pero nos puede dar un poco más de detalle sobre los empleados que han estado haciendo uso de la herramienta.

1 - Módulo de Clientes

Esta es la primera opción que aparece en el menú lateral y de hecho es el punto de partida de toda esta herramienta. Para rellenar correctamente el resto de módulos hay que tener clientes registrados en la aplicación para poder asociarlos a ellos. De todas formas no es obligatorio crear un cliente para poder crear registros de otros módulos, más tarde podría editarlos y asignarles el cliente.

Relaciones: El módulo de clientes está relacionado con vehículos, servicios, citas, recordatorios, presupuestos y facturas (Es el módulo que más relaciones tiene ya que es uno de los principales.) si nos fijamos en la parte inferior podemos observar esos subpaneles los cuales nos permiten ver las relaciones que tiene y acceder a ellas directamente. También gracias a los botones de la derecha podremos editar el registro y eliminarlo respectivamente.

2 - Módulo de Vehículos

Una vez dados de alta los clientes podemos acceder al módulo de Vehículos, en éste módulo como bien dice el nombre podremos dar de alta los diferentes vehículos con la información respectiva a la matrícula, marca, tipo de motor, color y numero de bastidor entre otros.

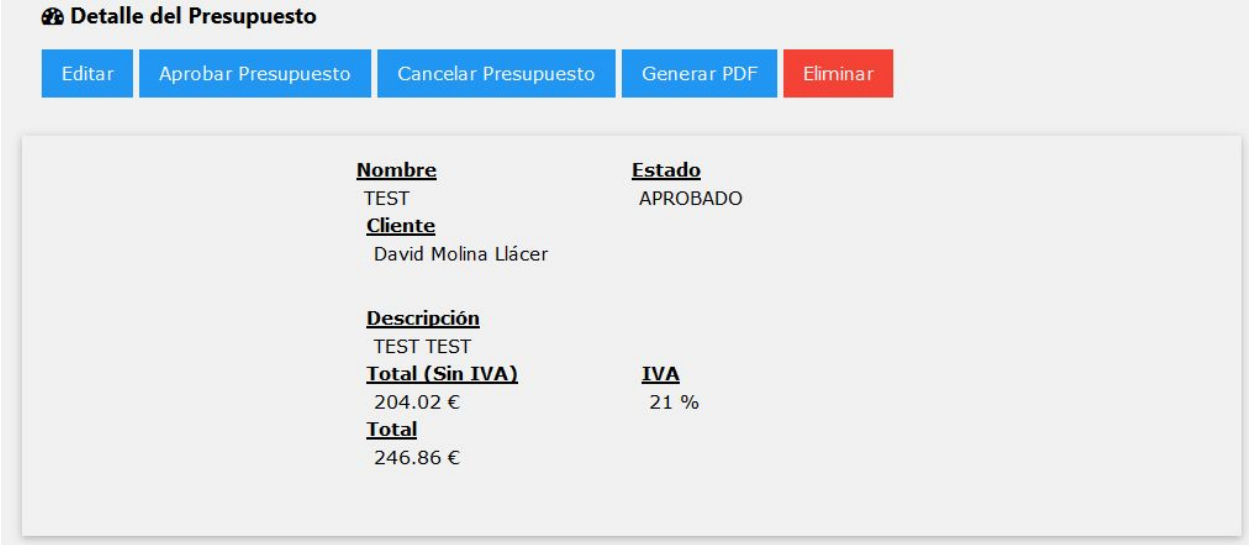
Relaciones: A través del módulo de vehículos podemos relacionarlo con el módulo de clientes y con el de servicios de forma que tenemos un historial de los servicios que se han realizado al vehículo.

3 - Módulo de Presupuestos

Una vez recogida la información del cliente y el vehículo podemos empezar a generar un presupuesto. Si no necesitase presupuesto por que es un servicio que se repite puede saltarse este paso. El módulo de presupuestos tiene en la página de edición una relación con el módulo de ítems el cual nos permite obtener los que ya previamente almacenados, de forma que si pulsamos sobre el botón **+** podemos ir seleccionando ítems una lista y obtener el precio. Una vez obtenido el precio, si marcamos la cantidad se nos auto calculará el precio total. Una vez establecido el total tendremos la opción de marcar el IVA, el cual nuevamente nos recalculara el total acorde a este.

Finalmente si el presupuesto es aceptado podremos pulsar sobre el botón **Aceptar Presupuesto** el cual lo marcará como aceptado y además nos llevará a la página de cita la cual hablaremos a continuación. En el caso de que tuviera que generar más adelante la factura bastaría con pulsar sobre el botón **Generar Factura** este botón nos volcará toda la información a una factura nueva en la cual podremos editar lo que necesitemos. También si queremos imprimir el presupuesto o enviarlo por correo electrónico, podemos generarlo en PDF para

facilitar la tarea pulsando sobre el botón **Generar PDF** el cual nos abrira en otra pestaña la cual nos renderiza todo el contenido en un PDF.

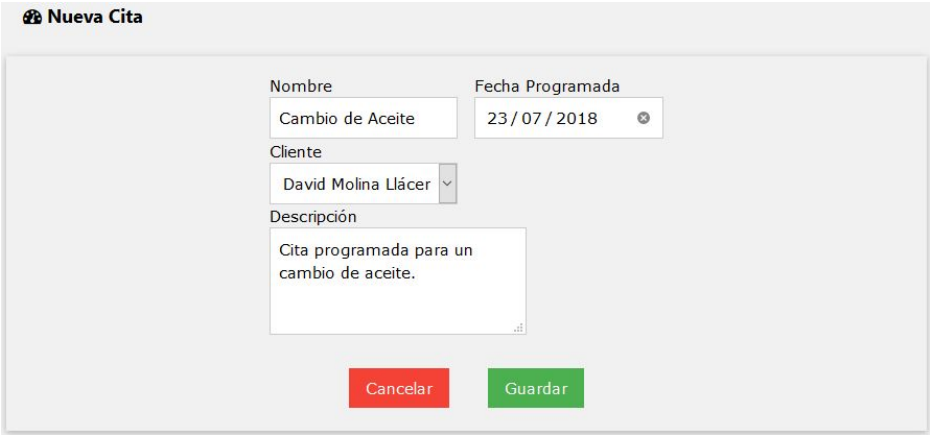


Relaciones: El módulo de presupuestos únicamente tiene 2 relaciones: Una es con el cliente a través de la enlace en la ficha de detalle (Si pulsamos sobre él nos enlazará con la información del cliente) y la otra relación es con ítems de forma que podemos ver los items asociadas a la factura.

4 - Módulo de Citas

Ahora es el momento de agendar una cita. Simplemente seleccionamos al cliente el cual queremos agendar una cita y le programamos una fecha. Este módulo al igual que el anterior no es obligado debido a que si es una reparación que se puede efectuar en el mismo dia no la necesitará. Por lo tanto si ese fuese el caso puede saltarse este proceso.

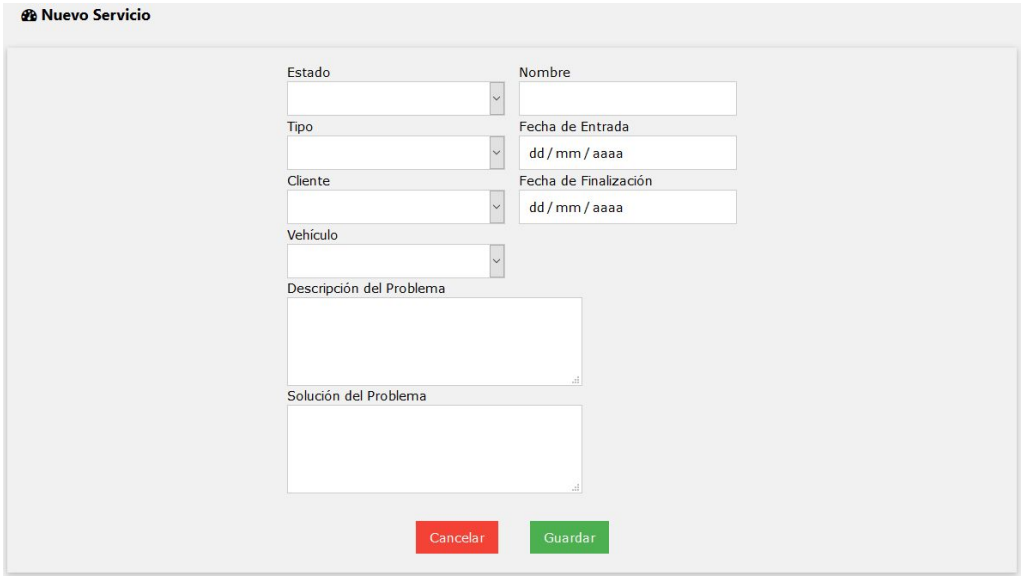
Una vez hemos generado una cita y esta ha llegado al día podemos ir a buscarla al listado de citas o bien desde la página principal ya que tenemos en el recordatorio las citas más recientes. Una vez localizada la cita, en la vista de detalle si pulsamos sobre el botón **Confirmar Cita** cambiará su estado a confirmada y nos abrirá la vista de edición del servicio el cual vamos a efectuar.



Relaciones: El módulo de presupuestos dado que la información que se proporciona es la del cliente únicamente veremos un subpanel de cliente el cual nos muestra la información.

5 - Módulo de Servicios

Este es uno de los módulos principales ya que todo se centra en la informatización del servicio que ofrecemos al cliente, es decir, el estado en el que nos llega un vehículo y las acciones que hacemos sobre él para su correcta reparación. Por ello los campos que deberemos rellenar son: el tipo de servicio, el estado actual del servicio (Nuevo, En Proceso, Finalizado), descripción del problema y la solución aplicada. Cuando tengamos el servicio en estado finalizado estamos listos para generar la factura a través del botón Generar Factura de la parte superior de la vista de detalle.



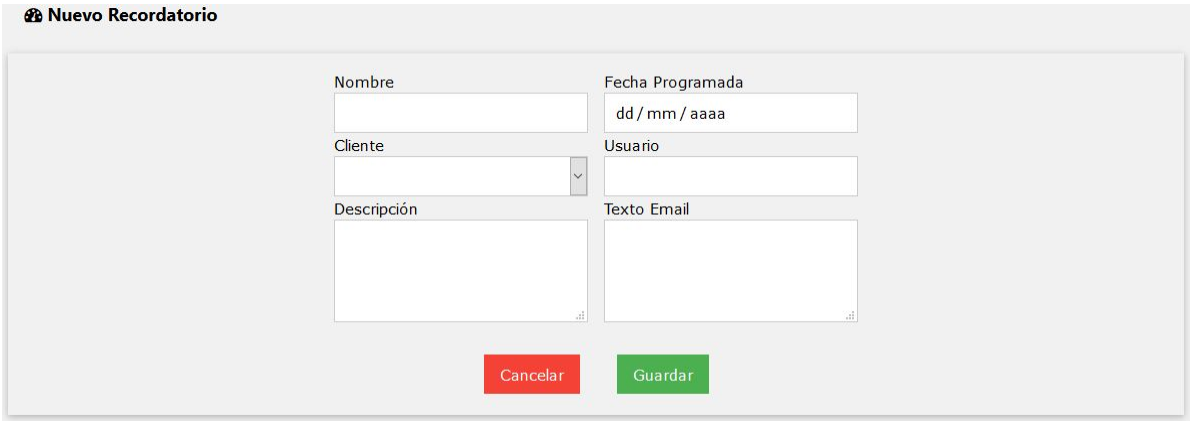
Relaciones: A través del módulo de servicio tenemos acceso a los subpaneles de cliente y vehículo.

6 - Módulo de Facturas

Tras haber completado el servicio estamos listos para generar la factura al cliente. El funcionamiento de este módulo es completamente igual al de presupuestos ya que comparten la misma estructura. Procederemos seleccionando los ítems implicados en el servicio para que obtenga el precio y calcule el total en función del número de unidades especificados en cada ítem.

7 - Módulo de Recordatorios

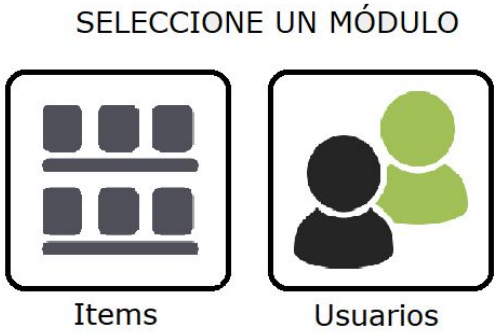
Finalmente tras cerrar todo el proceso, si el servicio ofrecido requiere una periodicidad este módulo es perfecto podemos informar al cliente en un futuro sobre un servicio que tenga próximo. Para ello establecemos la fecha en la que queremos que se produzca el aviso junto con un nombre identificativo y una descripción para conocer más el detalle del próximo servicio. Hecho esto y guardado el recordatorio, la semana de antes se hará un envío de email al cliente para saber que tiene que contactar al mecánico y también se enviará el email al mecánico de forma que él también pueda prever un servicio que le va a llegar próximo o incluso llamar al cliente para agendar una cita. Para ello se dispone de un botón en la parte superior de la vista de detalle con el nombre de **Agendar**. Tras pulsar el botón se nos preparará el formulario de la cita para establecer día y así queda cerrado el bucle de los recordatorios.



Acceso como Usuario Administrador

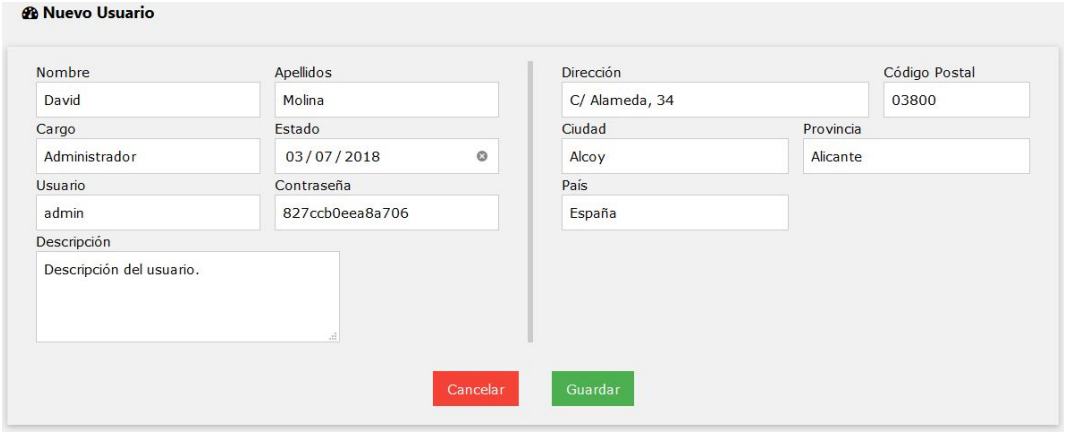
Si el usuario con el que accedemos tiene el rol de administrador tendremos acceso a más módulos implicados en este proceso además de todos los mencionados anteriormente. Para ello una vez identificados, al tener permisos de administrador se nos mostrará el icono: ⚙

Este icono nos desplegará un menú de acceso a dos módulos diferentes. Uno de usuarios y otro de ítems.



Módulo de Usuarios

Mediante este módulo podemos gestionar los usuarios que acceden a esta herramienta. Este módulo presenta el mismo sistema que los módulos anteriores, es decir, una lista de usuarios donde podemos crear, editar o eliminar usuarios. Podremos asignarles los roles determinados (Administrador y Mecánico) a través los cuales se le bloqueará ciertos accesos a la aplicación y además cambiar la contraseña de acceso cuando sea necesario.



Módulo de Items

Este módulo nos permite añadir toda la lista de ítems disponibles para su uso en las facturas y los presupuestos. Cada ítem puede ser de dos tipos, Articulo y Servicio. Artículo es referido a cualquier pieza material que se le añada al coche y el servicio es la acción que toma el mecánico para hacer la reparación. Una vez dados de alta aparecerán en el desplegable de ítems del módulo de presupuestos y facturas.

4.3 Estadísticas de explotación

4.3.1 Estudio comparativo

Ahora que la aplicación está llegando a su final es momento de plantearse qué puede ofrecer esta aplicación con respecto al resto de herramientas del mercado. De esta manera podemos hacernos una idea del interés que puede llegar a generar esta herramienta y además establecer unos límites sobre lo que puede hacer o no puede hacer para que en futuras líneas de desarrollo se tenga en cuenta el desarrollo de esa funcionalidad.

Para el estudio comparativo voy a usar las aplicaciones que estudié al inicio del proyecto y sobre las que basé mi aplicación ya que busqué las herramientas más usadas del mercado. Estas son herramientas profesionales aplicadas hoy en día en el mundo profesional por lo que la competencia ya es dura nada más comenzar. De todas formas es una buena manera de obtener una idea real y me permitirá saber que he mejorado con respecto a ellas o incluso que me queda mejorar ya que dado que acaba de arrancar todavía le queda un largo camino.

Tallermatic

Esta es una aplicación de escritorio con una interfaz muy antigua, además el punto más fuerte sobre lo que está basado todo es la facturación. A diferencia de mi proyecto, es una aplicación web por lo que la información es accesible desde cualquier lugar y prácticamente en tiempo real. Otro punto a favor es la interfaz de usuario, la herramienta que he desarrollado ofrece una interfaz más moderna, sencilla y cómoda de utilizar. Además dado que son interfaces basadas en HTML permite la modificación a lo largo del tiempo a través de una serie de temas distintos. También mi aplicación es de diseño adaptable por lo que permite el uso de dispositivos móviles para hacer uso de ella rápidamente en el taller. Aunque es verdad que Tallermatic poseé una aplicación móvil, ésta sólo funciona en sistemas operativos android y ofrece funcionalidad limitada. En mi caso dado que el acceso es por web puedes acceder a toda la herramienta sin restricción alguna.

TallerGP

Ahora toca movernos sobre otra aplicación en este caso mucho más moderna y más completa. TallerGP es una aplicación que se presenta como un duro rival. En este caso hablamos de una aplicación web igual que la mía por lo que ofrece de igual manera el acceso remoto al gestor y con la información siempre al dia. La interfaz a pesar de ser web se ha basado en aplicación de escritorio probablemente porque esté desarrollada en Java o .NET. En

cuanto a funcionalidad he de decir que esta aplicación es algo más completa ya que ofrece funcionalidades que por el momento son carentes en mi aplicación. Toda la parte de gestión de clientes, vehículos, servicios, citas y facturación está en ambas pero quizás han sabido añadir algún atractivo más como pueda ser el envío de SMS y gestión de coches de sustitución. De todas formas Son pequeños añadidos que tendré en cuenta para futuras versiones de la aplicación.

GesTalleres

Ésta aplicación se nos presenta de la misma forma que la primera aplicación con la que comparé. En primer lugar vemos una aplicación de escritorio sencilla al igual que el diseño, el cual es muy sencillo y con una apariencia más bien antigua. Esta aplicación está más enfocada a la facturación ya de control de vehículos hay más bien poco. También cabe destacar que esta aplicación en cuanto a uso en dispositivos móviles carece de alguna aplicacion Android o iOS a través de la cual poder consultar la información remotamente. Esto además se suma a que la información entre distintas aplicaciones no se mantiene completamente sincronizada sino que hace uso de un solo sistema en el cual están todos los datos. Haciendo comparación con mi aplicación, esta me permite mantener centralizada toda la información en un servidor a la que es accedida a través de cualquier ordenador o dispositivo móvil. Gracias a esto cada uno puede hacer uso de la información independientemente y además completamente actualizada.

Tras este pequeño estudio podemos observar las ventajas que aporta mi aplicación con el resto. Por ello no significa que las demás sean peores que ésta, sino que ofrecen otras pequeñas funcionalidades que pueden resultar atractivas al cliente y de las que aprenderé para poder desarrollarlo en esta. De esta manera podré ofrecerle al cliente un mejor producto.

5 Conclusiones

5.1 Conclusiones personales

La realización de este proyecto ha sido una oportunidad muy buena para ampliar mis conocimientos sobre la programación web. Al principio fué un poco confuso establecer el alcance de este proyecto debido a que es inmensa la cantidad cosas que se pueden desarrollar para un taller mecánico, a medida que se desarrolla algo siempre piensas en dos cosas más que se podrían implementar. Gracias a una charla con el tutor me ayudó a poner las ideas en orden y tomar un punto de partida sobre la que ir construyendo poco a poco este trabajo.

Al principio se me presentó una gran duda sobre que metodo iba a utilizar para desarrollar el trabajo en PHP. Existen multitud de frameworks que me hubiesen ayudado a realizar el trabajo una base muy sólida, un problema fue hacer la elección de uno de ellos además de invertir cierto tiempo en dominar ese Framework. Tras pensarlo exhaustivamente me plantee la posibilidad de desarrollar uno propio de forma que este proyecto además de servir como desarrollo de una aplicación web, también me sirviera para conocer cómo están diseñados por dentro y lograr un código desarrollado completamente por mi. Finalmente me decanté por la segunda opción, gracias a ello he podido desarrollar algo que además me sirve para futuras aplicaciones permitiendo mantener mi propio código.

Tras resolver las diversas complicaciones que suponía crear un Framework en PHP para el lado del servidor, ahora se presentaba un nuevo problema, esta vez en el lado del cliente y con un nuevo lenguaje, Javascript. Javascript era un lenguaje que había utilizado anteriormente pero para realizar tareas básicas como alguna automatización de campos. Tras realizar algunas pruebas y gracias a la ayuda de JQuery pude realizar una parte importante en la aplicación, en este caso es la comunicación mediante AJAX.

A medida que ya estaban las funcionalidades implementadas se presentaba un nuevo problema y era el diseño de las páginas web que desarrollaba. Anteriormente ya había realizado algún trabajo en HTML y CSS por lo que dominaba esta parte de la materia pero el diseño siempre se me ha dado muy mal por lo que corría el riesgo de tener una buena base pero una presentación pésima. En este punto me ayudó mucho un Framework de CSS el cual ya tenía un diseño por defecto y me permitió centrarme más en la programación y lograr un diseño de páginas web muy bueno además de venir con un diseño adaptable.

Finalmente unificando todo el conocimiento adquirido de varios lenguajes, arquitecturas y diseños he logrado crear esta aplicación para la gestión de los talleres mecánicos.

5.2 Futuras líneas de desarrollo

Tras el trabajo realizado en esta aplicación siempre se me ocurren nuevas ideas o mejoras que habría que implementar para mejorar la experiencia del usuario:

1. A medida que ha ido creciendo la funcionalidad se han ido implementando más módulos ello implica que cada vez la lista en el menú de la aplicación sea más extensa. Esto provocará que sea cada vez más difícil encontrar el módulo por lo que el cambio de ventanas se verá entorpecido. Lo ideal seria ver la manera de agrupar por funcionalidad de forma que sea más fácil de encontrar.

2. Dentro de la vista de detalle, los subpaneles en los que aparecen las relaciones con los demás módulos deberían de mostrar un botón para crear el registro al vuelo y no tener que ir a dicho módulo para editar.

3. A la hora de crear un nuevo registro si queremos asignarle un vehículo por ejemplo hemos de ir al módulo de vehículos y crearlo para después poder asignarlo. Se ha de poder crear nuevos registros mientras se asignan de manera que no se pierda nunca el formulario que se estaba rellenando.

4. Expandir la información que tiene la aplicación de cada módulo. Por el momento cada módulo recoge datos básicos los cuales con ayuda de algún mecánico interesado se pueda personalizar todavía más.

5. Como futura línea de desarrollo se propone una página intermedia accesible a través del cliente para registrar un cita.

6. También al efectuar el pago, hacer a través de una conexión segura cifrada.

7. Extender roles de usuarios para limitar justo lo que el usuario ha de ver. Por ahora o eres Administrador o no. O todo o nada. Deberia haber mas ajustando la visualización de los registros.

6 Bibliografia

LIBROS

- PHP and MySQL Web Development - Luke Welling & Laura Thomson
- Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites - Robin Nixon
- Programming PHP - Rasmus Lerdorf

PÁGINAS WEB

- http://www.php.net
- http://www.w3schools.com
- https://developer.mozilla.org/es/docs/Learn/JavaScript
- http://www.comocreartuweb.com/curso-de-html
- https://api.jquery.com/
- https://davidburgos.blog/como-hacer-pruebas-de-estres-tu-servidor/
- http://httpd.apache.org/docs/2.2/programs/ab.html
- es.slideshare.net/carsanta/nucleo-4-diseo-de-db-con-modelo-entidad-relacin
- https://www.hostinger.es/tutoriales/conectar-php-mysql/

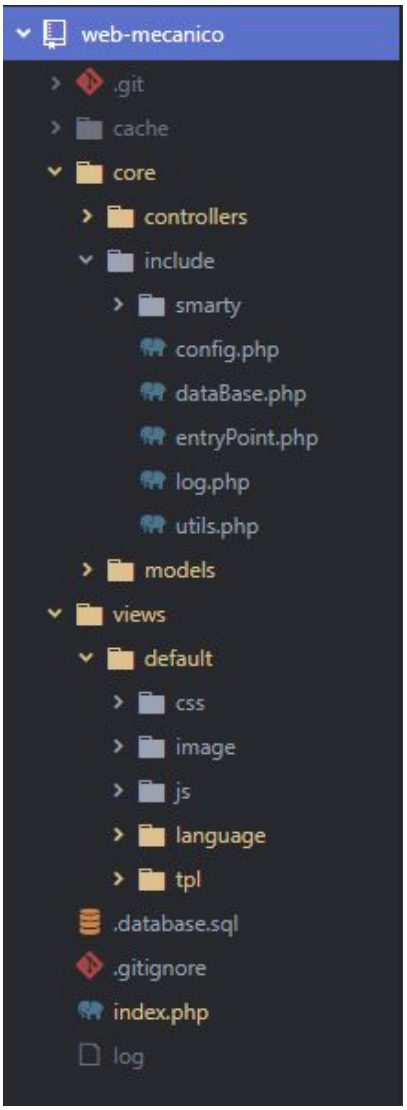
7 Acrónimos

- **PHP**: Acrónimo recursivo que significa PHP Hypertext Preprocessor.
- **HTML**: HyperText Markup Language.
- **CSS**: Cascading StyleSheets.
- **JSP**: JavaServer Pages.
- **MVC**: Modelo-Vista-Controlador.
- **SQL**: Structured Query Language.
- **JSON**: JavaScript Object Notation.
- **XML**: eXtensible Markup Language.
- **SMS**: Short Message Service.
- **IP**: Internet Protocol.
- **AJAX**: Asynchronous JavaScript And XML.
- **DB**: Database.
- **ISAM**: Indexed Sequential Access Method.
- **CRUD**: Create, Read, Update and Delete.
- **DML**: Data Manipulation Language.
- **GUI**: Graphic User Interface.
- **API**: Application Programming Interfaces
- **IVA**: Impuesto sobre el Valor Añadido.
- **FTP**: File Transfer Protocol.
- **CPU**: Central Processing Unit.
- **POO**: Programación Orientada a Objetos.
- **RAM**: Random Access Memory.
- **URL**: Uniform Resource Locator.

8 Anexos

Para el desarrollo de esta aplicación ha sido desarrollado un framework por mi, de forma que he podido crear, editar y comprender el código base sobre el que se basa cualquier framework. A continuación expondré las partes que lo componen junto con el código fuente para dejar claro el funcionamiento del mismo.

8.1 Estructura de Directorios



+ Cache: En esta carpeta se almacena cualquier contenido generado por las librerías que hay activas en este framework. De esta manera el siguiente uso es más rápido ya que no tiene que generar más código.

+ Core: En esta carpeta se almacena toda la lógica del framework. Es decir: ficheros de configuración (Aplicación y Base de Datos), librerías externas y las carpetas implicadas en el patrón MVC. (Controllers y Models)

+ Views: Esta carpeta contiene toda la capa pública que será enviada al usuario (HTML, CSS, JavaScript y Imágenes). Está preparada para almacenar un conjunto de vistas para que más adelante pueda ser utilizado a moto de temas y poder cambiar el aspecto visual según la decisión del usuario.

+ .database: Es una copia limpia preparada para hacer una instalación de la aplicación desde 0. Contiene todas las tablas implicadas y preparadas para subir a modo de consulta en la base de datos SQL.

+ index: Este es el fichero por defecto que el servidor cargará y este es el punto de partida de la aplicación.

+ log: Este es un fichero generado por una clase Log que me permite volcar los eventos que ocurren en la aplicación a medida que el usuario hace uso de ella.

8.2 Inicio de la Aplicación

El punto de partida se encuentra en el fichero **index.php**, esto es así porque por defecto es el fichero que el servidor cargará cuando el usuario haga una petición.

Dado que es el fichero principal lo primero de todo es cargar toda la configuración de la que vayamos a hacer uso. De eso se encarga el siguiente fragmento de código

```
if(file_exists('core/include/entryPoint.php')){
   include 'core/include/entryPoint.php';
}else{
   die();
}
```

Esto nos cargará el código albergado en el fichero **entryPoint.php**. Este fichero se encargará de cargar los datos de configuración de la base de datos y de la propia herramienta tales como el idioma por defecto, el tema que se va a utilizar, funciones de las que hace uso la aplicación y las rutas que emplearán ciertas librerías para volcar sus ficheros. Todo esto lo consigue mediante el código mostrado a continuación.

```
 if(file_exists('core/include/config.php')){
   include 'core/include/config.php';
}else{
   die();
}

if(file_exists('core/include/utils.php')){
   include 'core/include/utils.php';
}else{
   die();
}

if(file_exists('core/include/log.php')){
   include 'core/include/log.php';
}else{
   die();
}

if(file_exists('core/include/smarty/Smarty.class.php')){
   include 'core/include/smarty/Smarty.class.php';
}else{
   die();
}
```

Todas estas inclusiones son imprescindibles para el funcionamiento de la aplicación, es por eso por lo que primero hace una comprobación para ver si existe el fichero antes de cargarlo. Si no encontrase el fichero la aplicación se detendría en el momento ya que si falta algún fichero de esos no tiene sentido que continúe la ejecución. Dentro de este fichero encontramos por último el fragmento de código siguiente:

```
if(!isset($_GET['lang'])){
  if(file_exists('views/' . $config['theme'] . '/language//' . $config['lang'] . '.php')) {
      include 'views/' . $config['theme'] . '/language//' . $config['lang'] . '.php';
  } else {
      $log->Log('ERROR', 'Fichero de idiomas no encontrado.');
      die();
  }
}else{
  if(file_exists('views/' . $config['theme'] . '/language//' . $_GET['lang'] . '.php')) {
      include 'views/' . $config['theme'] . '/language//' . $_GET['lang'] . '.php';
  } else {
      if(file_exists('views/' . $config['theme'] . '/language//' . $config['lang'] . '.php')) {
         include 'views/' . $config['theme'] . '/language//' . $config['lang'] . '.php';
      } else {
         $log->Log('ERROR', 'Fichero de idiomas no encontrado.');
         die();
      }
  }
}
```

En este fragmento de código dado que hemos cargado los ficheros de de configuración ahora ya podemos acceder a la información. Para ello lo llamamos con la variable **$config['lang']**, esta es una variable almacenada en un array con toda la información. Para hacerse una idea de las variables de las que está compuesta el array de configuración el código alojado en **config.php** es el siguientes.

```
$config = array(
    'lang' => 'es',
    'theme' => 'default',
    'cache' => 'cache',
    'address' => 'http://' . $_SERVER['SERVER_NAME'] . $_SERVER['PHP_SELF'],
    'db' => array(
        'host' => 'localhost',  'user' => 'root',  'password' => '',  'db_name' => 'imechanic'
    ),
    'log' => array(
        'name' => 'log',
        'ext' => ''
    )
);
```

Finalmente ahora que ya tenemos todos los ficheros de configuración y las librerías cargadas ya podemos inicializar los objetos de las clases que nos permitirá mostrar plantillas por pantalla y imprimir datos en el fichero de log. Más adelante accederemos a ellas mediante la variable $GLOBALS de PHP.

```
$log = new log($config['log']['name'], $config['log']['ext']);
$smarty = new Smarty();
```

8.3 Función principal de la Aplicación

Ahora que ya hemos cargado todos los ficheros necesarios para hacer funcionar la aplicación y además hemos inicializado los objetos, ya puede cargarse la función principal de la aplicación. Lo puntos principales de la función es:

1. Revisar si ya existe una variable de sesion con la información de usuario. Esto es fundamental para continuar revisando los demas parametros ya que de no tener variable de sesion hará caso omiso de los parámetros y te mostrará la página de acceso desde donde se tendrá que identificar para que ya si que genere una id de sesión.
2. A continuación, si tiene variable de sesión lo que hará es comprobar el parámetro “page” que se le pasa a través de la URL. A través de ese parámetro comprobará si existe un controlador con ese nombre, en caso afirmativo cargará el controlador.
3. Una vez se ha cargado el controlador, esté llamará al método común en todos que es Show(). En caso contrario nos conduciremos al usuario a la página principal ya que esa es propia de la herramienta y sabemos que siempre existirá.

```
// Comprobamos si se ha iniciado la sesión para permitirle ir más allá
if(isset($_SESSION['id'])){
   // Determinamos qué páginas han de ser cargadas según parámetros.
   if(isset($_GET['page']) && file_exists('core/controllers/controller.' . $_GET['page'] . '.php')){

        include 'core/controllers/controller.' . $_GET['page'] . '.php';
        $page = 'CR_' . $_GET['page'];

        $controller = new $page();

        if(isset($_GET['action']) && method_exists($controller, $_GET['action'])) {
            $metodo = $_GET['action'];
            $controller->$metodo();
            // Cambio de Version, ya no puede ser usado.
```

```
                //$controller->$_GET['action']();
            } else {
                $controller->Show();
            }
    } else {
        if(file_exists('core/controllers/controller.main.php')) {
                include 'core/controllers/controller.main.php';
        } else {
                $GLOBALS['log']->Log('ERROR', 'Fichero "controller.main.php" no encontrado.');
                die();
        }

        $controller = new CR_main();
        $controller->Show();

    }
} else {
    include 'core/controllers/controller.users.php';

    $controller = new CR_users();

    if(isset($_POST['user']) && isset($_POST['password'])){
        $controller->Login();
    } else {
        $controller->ShowLogin();
    }
}
```

8.4 Controlador de la Aplicación

Los controladores están ubicados dentro de la carpeta “controllers” dentro del “core” del framework. Cada uno de los controladores representa a un módulo y dentro están los métodos los cuales invocamos para realizar ciertas acciones sobre los registros de ese módulo tales como listar, editar y crear registros. Todo esto ayudado del “modelo”, una clase la cual nos servirá para comunicarnos con la base de datos.

Dentro del fichero encontraremos la clase las cuales van precedidas con el prefijo **CR_**. Todo ello para identificarlas mejor dentro del código. Una vez dentro de la clase lo primero es cargar el fichero del “modelo” para que más adelante nos ayude con la base de datos. Para ello haremos uso del constructor de la clase de manera que cuando se inicialice nos cargue toda la información necesaria.

```
        function __construct() {
            if(file_exists('core/models/model.<Nombre_Modulo>.php')) {
                include 'core/models/model.<Nombre_Modulo>.php';
            } else {
                $GLOBALS['log']->Log('ERROR', 'Fichero "model.<Nombre_Modulo>.php" no
encontrado.');
                die();
            }
        }
```

Ahora que hemos cargado los ficheros necesarios ya podemos acceder al resto de métodos para ejecutar la tarea a demandar. Como hemos visto antes el método a ejecutar es pasado a través del parámetro 'action'. De forma que podemos extender la clase con los métodos que sean necesarios ya que tenemos una forma externa de llamarlos.

Ahora mismo en este framework son distinguibles dos tipos de métodos:

1. Métodos de Visualización: Estos métodos nos permiten recoger la información del resto de métodos de soporte y devolver una página web que será renderizada a través de la librería de plantillas Smarty. Estas plantillas son recogidas de la carpeta de views la cual representa la parte de la Vista dentro del patrón MVC. Estos métodos son:

    a. **Show()** -> Hace una visualización de la vista de lista.
    b. **ShowEdit()** -> Nos permite acceder a la edición del registro.
    c. **ShowDetail()** -> Con esta visualizamos el detalle del registro.

    Cada una hace referencia a las distintas vistas de la aplicación que aparece en el apartado de diseño del producto.

2. Los de Soporte: Estos métodos principalmente sirven de ayuda a los de visualización. Son los encargados de comunicarse a través del "modelo" con la base de datos y ofrecer la información para que pueda ser visualizada. A pesar de ello no quiere decir que no puedan servir para otra cosa ya que si se crea un método que devuelva una resultado en JSON, este puede ser llamado independientemente de la vista por una aplicación externa. Los principales métodos básicos que los componen actualmente son:

    a. **newBean()** -> Esta sirve de ayuda a la vista de edición para generar el registro nuevo.
    b. **getBeanList()** -> Este método nos ayuda a obtener la lista de registros para mostrar el listado en la vista correspondiente.

c. **getBean()** -> Obtenemos el valor de un único registro para ser mostrado en la vista de detalle.
d. **deleteBean()** -> Nos permite eliminar el registro seleccionado.
e. **getRelatedBeans()** -> Con este método obtenemos la información de cualquier módulo relacionado para ser mostrado en subpaneles.

8.5 Modelo de la Aplicación

Una vez el controlador está cargado, mientras efectúa la lógica de la aplicación este se sirve de la base de datos para almacenar y recuperar la información. Esto lo realiza mediante los modelos. Al igual que los controladores hay uno por módulo y a pesar de que normalmente el controlador de determinado módulo suele llamar al modelo del mismo módulo, no tiene porqué ser siempre así.

Lo primero de todo es declarar la clase que lleva el nombre de módulo y lo heredamos de la clase database. Esta es una clase en la cual englobamos la información de acceso a la base de datos y mediante métodos hacemos la conexión y desconexión así como las consultas. Todo ello haciendo uso de la clase mysqli que incluye PHP.

Los métodos que hay desarrollados aquí usan el mismo nombre que muchos de los métodos de los controladores. Eso es porque los son nombres descriptivos y para que el controlador haga una tarea el 80% de veces requiere acceso a la base de datos. Es por eso por lo que el método del controlador tendrá el mismo nombre que el método del modelo que se encarga de la misma función. Entre otros, los métodos usados en el modelo normalmente son:

1. **newBean**() -> Este método junto con el método del mismo nombre del controlador permite crear nuevos registros. En este caso, el modelo es el que genera la consulta SQL (INSERT INTO o UPDATE) a través de la información que el controlador le pasa.
2. **getBeanList**() -> Esta es una consulta SQL (SELECT) en la que listamos todos los registros para ser mostrados en la vista de lista.
3. **getBean**() -> Este nos devuelve un único registro a través del identificador que es transmitido por el controlador. Para ello se utiliza una consulta SQL (SELECT) pero filtrada. (WHERE)
4. **deleteBean**() -> Con este método podemos borrar los registros tal y como su nombre indica. Este método no genera la típica consulta de eliminar en SQL (DELETE), sino que realiza una actualización del registro (UPDATE) a través la cual cambia el valor del campo “deleted” a 1.

5. **login**() -> Este es un modelo utilizado en el módulo de usuarios. Es un ejemplo de un método independiente el cual no comparte nombre con ningún método del controlador del usuario pero en otros métodos más especifico cuando necesita saber si la identificación es correcta hace uso de este método.

Vistas de la Aplicación

Las diferentes vistas que forman la aplicación están almacenadas en formato de plantilla (.tpl). Están ubicadas dentro de la carpeta default que está a su vez dentro de views. Views puede contener tantas carpetas como se quieran y representan los diferentes temas que puedan haber. Para cambiar el tema basta con especificar el nombre de esa carpeta en el fichero de configuración, así cada vez que cargue el contenido usará los ficheros de esa carpeta especificada. Para ello debemos mantener la misma estructura de directorios los cuales están compuestos por:

1. **CSS** -> Es la carpeta que contiene las hojas de estilo de las que el tema pueda hacer uso.
2. **IMAGE** -> Cualquier imagen que vaya a ser mostrada irá siempre almacenada en esta carpeta. Dentro se pueden generar directorios para mantenerlas organizadas.
3. **JS** -> Cualquier fichero con contenido en JavaScript ha de ser alojado en este directorio.
4. **LANGUAGE** -> En este directorio van los distintos ficheros de idioma que van a ser incluidos dentro de esas páginas. Están compuestos por arrays en los cuales la clave es la etiqueta que será común en todos los ficheros de diferentes idiomas y el valor que es el texto con la traducción que corresponde.
5. **TPL** -> Esta carpeta contiene las plantillas con el contenido HTML de las cuales se alimentan los controladores y explotan a través de la librería Smarty.