Proyecto Final

Ciclo Formativo de Grado Superior
Desarrollo de Aplicaciones Web

(***Departamento de Informática***)

Gestoria Portol

Fecha: 9 de JUNIO de 2017

Autor: CRISTIAN DÍAZ PORTERO (43185091S)

Teléfono: 656 96 88 62

E-mail: cristiansmx2a@gmail.com

Tutor del proyecto:
Carlos Sogorb Valls

Centro:
IES Son Ferrer (Calvià – Mallorca)

Tabla de Contenido

1 - Introducción

En un principio la idea del proyecto era crear una aplicación para asignar rutas de reparto o recogida a camiones y verificar que la ruta se haya completado correctamente, pero tras investigar la manera de implementar la creación de rutas con la api de google, me di cuenta que se me iba de las manos el poder realizar lo que tenía en mente para la aplicación. Aparte aun pudiendo implementar las rutas existía el problema de necesitar estar la aplicación constantemente en primer plano para poder capturar la señal del gps a través del dispositivo(móvil o tablet), cuando la manera mas optima seria teniendo la aplicación trabajando en segundo plano en la plataforma nativa del dispositivo.

Por estas razones decidido cambiar el proyecto a una aplicación más orientada al temario dado en clase.

Durante el tiempo que estuve intentado avanzar el anterior proyecto, deje una base la cual me sirvió para el actual proyecto, aunque a medida que he ido avanzando, se ha refinado mucho más esa base.

1.1 - Justificación del proyecto:

Trabajo a media jornada en una gestoría en la cual se guarda el historial de traspasos y matriculaciones en papel. De ahí surgió la idea de trasladar todos los futuros datos de traspasos y matriculaciones a una aplicación dedicada a guardar este tipo de información y poder imprimir en caso de necesidad todos los registros deseados.

2 - Objetivos del proyecto:

La aplicación tiene como objetivo agrupar toda la información posible en un solo lugar, haciendo más fácil la búsqueda de datos. Como extra, se ha implementado la opción de exportar los datos deseados a una cierta cantidad de formatos, entre ellos PDF, EXCEL y JSON. Este último permite la automatización de migración de datos a aplicaciones externas.

Como objetivos personales, he logrado entender y controlar el funcionamiento del MVC a partir de un ejemplo básico, he creado mi propia versión intentando usar los justo y necesario para adaptarse a mi proyecto. Aunque parezca que el resultado final de la aplicación ha sido básico y se ha realizado poco esfuerzo, el esfuerzo ha residido más en conseguir un entorno donde poder programar cualquier aplicación teniendo un mínimo de conocimiento en un MVC. Y ademas esta aplicacion sera la base de una aplicación mucho más grande la cual irá creciendo a medida que la gestoría tenga mayor necesidad de trasladar su entorno de trabajo en papel a un entorno de trabajo totalmente informatizado.

También tengo que añadir que he conseguido implementar la aplicación en mi ordenador con visibilidad a internet, configurando redirección de puertos en el router, vinculando mi ip dinamica a una aplicación que trabaja con Dynamic DNS, y habilitado el acceso a través de apache.

3 - Técnicas y herramientas utilizadas

3.1 - Servidor web

Para montar el servidor se ha usado una arquitectura AMP, montada sobre windows usando el paquete de software WAMP, he optado por este paquete porque lo he usado en las prácticas, me ha parecido muy cómodo de configurar y me ha gustado mucho la manera de debugear los errores dados por php, detallando toda la ruta de ejecución de funciones hasta el error.

3.2 - IDE

Como entorno de desarrollo integrado(IDE) empecé utilizando eclipse, lo usaba en la empresa de prácticas y me familiarice con el, en algunos aspectos es cómodo como el hecho de integrar muchas herramientas en un solo entorno. Pero al final opte por usar sublime text para programar, ya que durante todo el curso he usado sublime y me desenvuelvo mucho mejor que en eclipse. La manera de editar el código sobre distintos lugares a la vez es muy cómoda y combinado con la posibilidad de seleccionar todos los resultados buscados y modificarlos a la vez es brutal.

Deje eclipse para otras tareas tales como el control de versiones, la visualización de ayuda de los métodos nativos de php, el sistema de buscador de palabras dentro del proyecto y la revisión de sintaxis.

3.3 - Lenguajes de programación

En un principio se tenía pensado utilizar javascript, php, html y css, y así fue durante un tiempo, pero dado que el proyecto no avanzaba al ritmo deseado, se comenzó a utilizar librerías para facilitar el trabajo. Primero se empezó con la librería DataTables(Hecha en JQuery), la cual al intentar configurar me abrió los ojos al potencial de JQuery y me enamoró de sus métodos de AJAX. A partir de aquí empecé a implementar BootStrap y otra librería llamada htmltable_export que iba a ser necesaria para poder exportar los datos de tablas en multitudes de formatos.

Para el control de versiones se ha utilizado el repositorio remoto público de github y todos los commits, push y fetch se han hecho a través de la extensión GIT de eclipse.
El repositorio remoto se llama como el antiguo proyecto, Rutas, y el propietario es kris90c4
Para visualizar los resultados en proceso de creación de la aplicación se ha usado la consola de chrome, y la ventana de incognito más la deshabilitación de la caché web, para evitar la ejecución de código antiguo

3.3 - Instalación de servidor

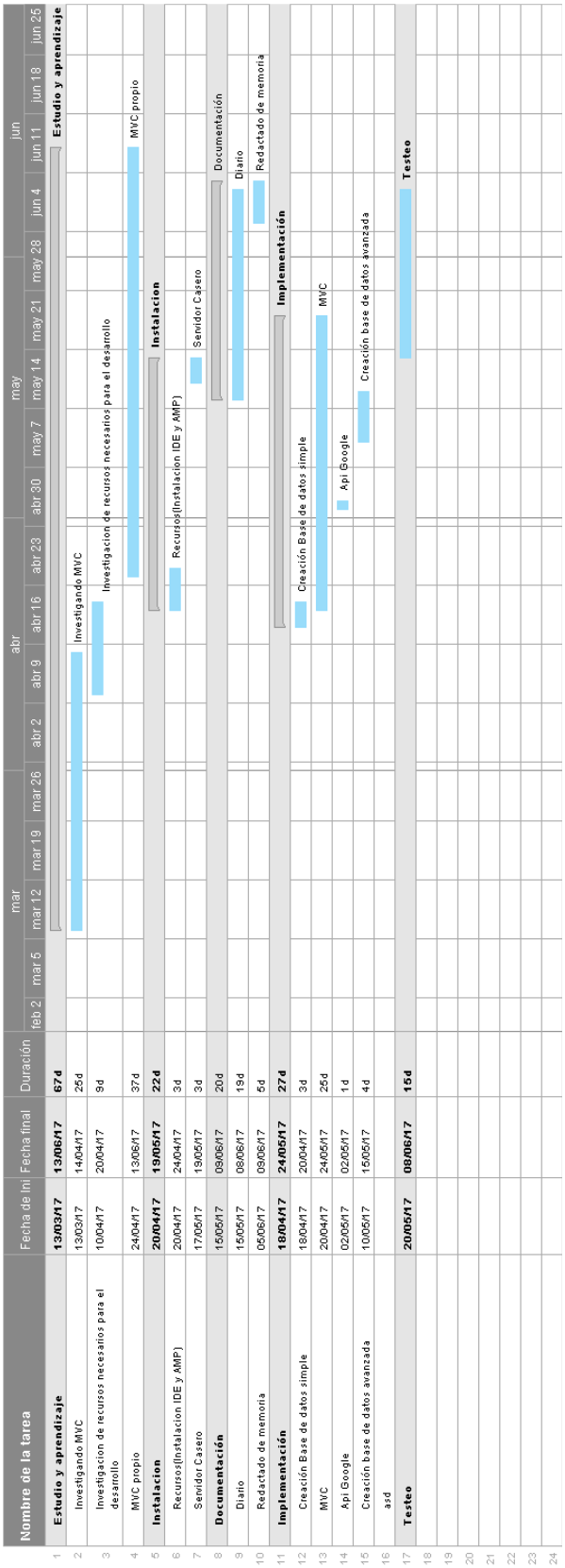
Para la creación del servidor se ha instalado el programa NO-IP el cual redirige mi IP dinámica actual a un dominio(gestoriaportol.ddns.net).
Después accediendo al router y se redirigió todas las peticiones sobre el puerto 80 a mi ip local. En el servidor apache se autorizó el acceso al directorio de mi proyecto a todo el mundo, y por último se añadió la excepción sobre el servicio de apache en el firewall de windows

3.4 - Especificaciones Servidor:

-Ancho de banda: 300 mb/s Subida / 300 mb/s Bajada
-Latencia: 18ms-30ms
-Procesador: i5 2500k 4.2GHz
-Ram: 8GB CL 6 1600MHz DDR3
-SSD: 128GB
-HDD: 9TB
S.O. Windows 10 Pro
Online: 8:00-24:00
Offline: 0:00-8:00
Tiempo Online: 16 horas
Host actual: **static-211-12-85-188.ipcom.comunitel.net**
Dominio: gestoriaportol.ddns.net

4 - Desarrollo de la aplicación

4.1 - Diagrama de Gantt

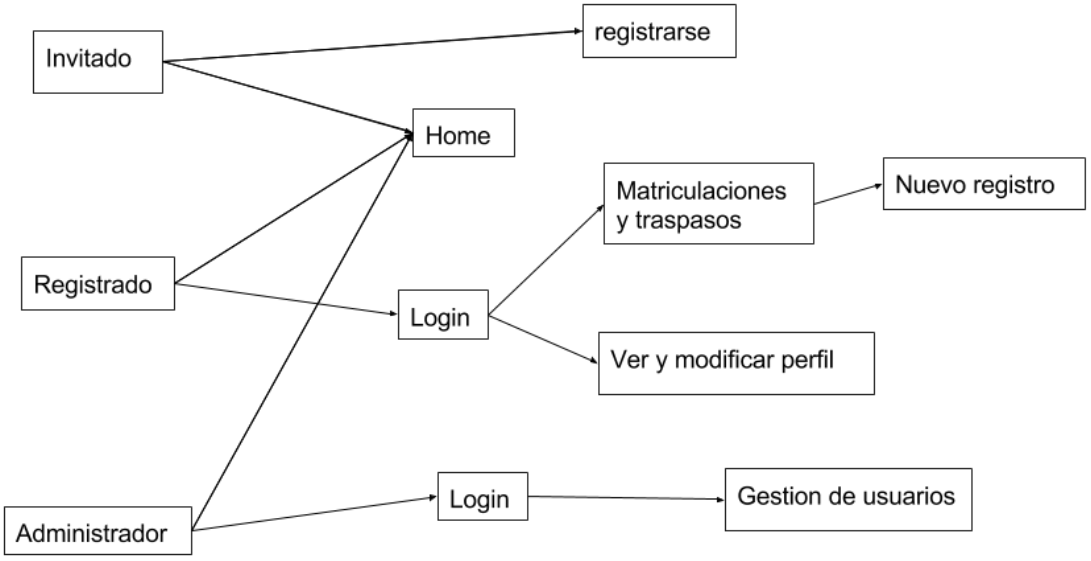


4.2 - Diagrama Casos de uso

Al tener un primer contacto con la aplicación en modo **invitado**, uno se encuentra con la posibilidad de visualizar la página home y/o registrar una nueva cuenta.

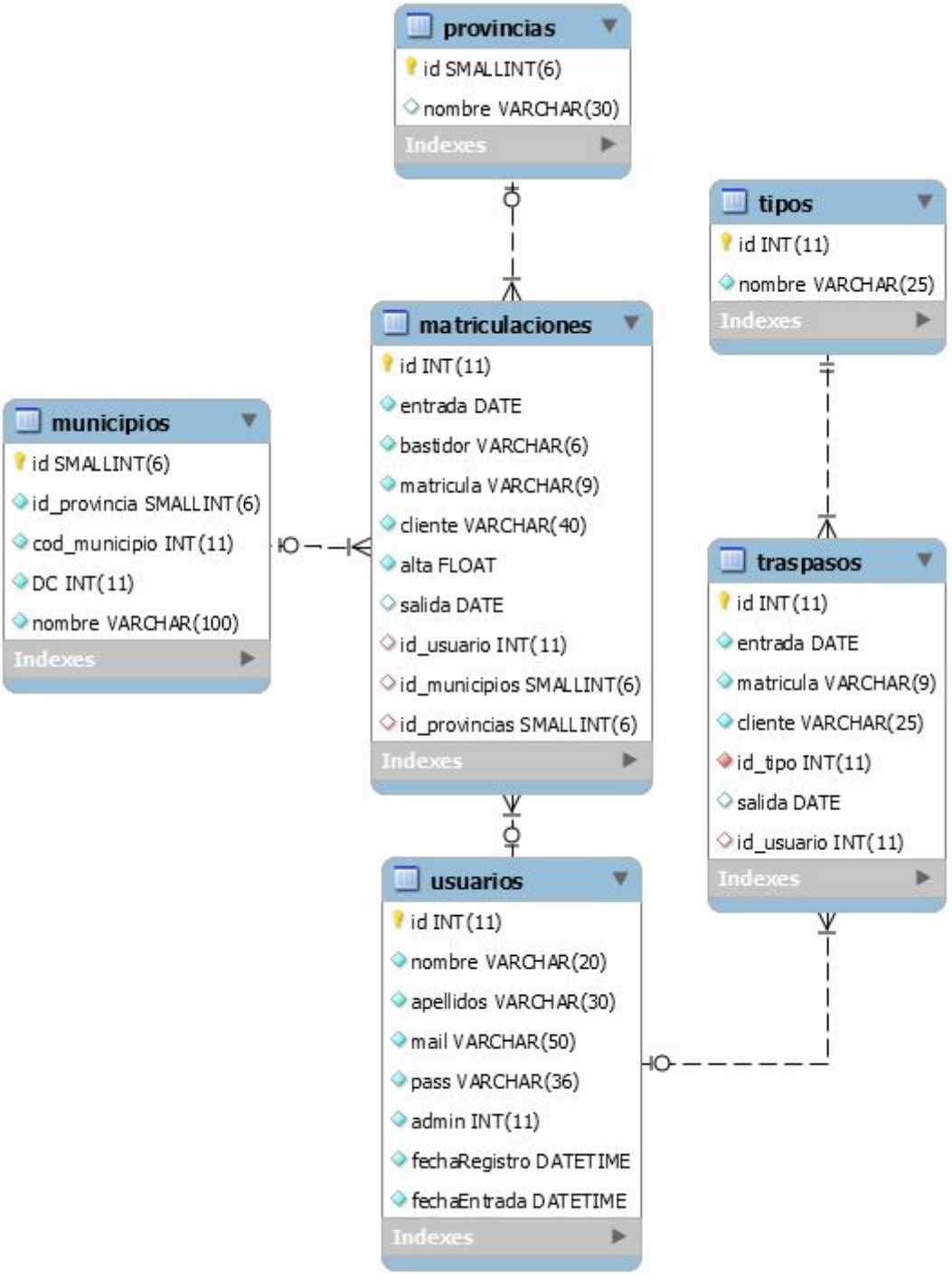
Una vez **registrado**, la próxima vez que se acceda a la aplicación, se dispondrá de la información necesaria como para usar el apartado de login y acceder a los servicios ofrecidos, los cuales incluye el acceso al perfil y el acceso a las tablas matriculaciones y traspasos y sus respectivos formularios de nuevo registro.

Un usuario **administrador** deberá loguearse al igual que un registrado, pero el contenido a ver sera unicamente su panel de gestión de usuarios, el cual le permite eliminar cuentas de usuarios y/o resetear sus respectivas contraseña. No se le restringe el acceso a los apartados de los registrados, pero tampoco se le facilita un link para realizar tales acciones.

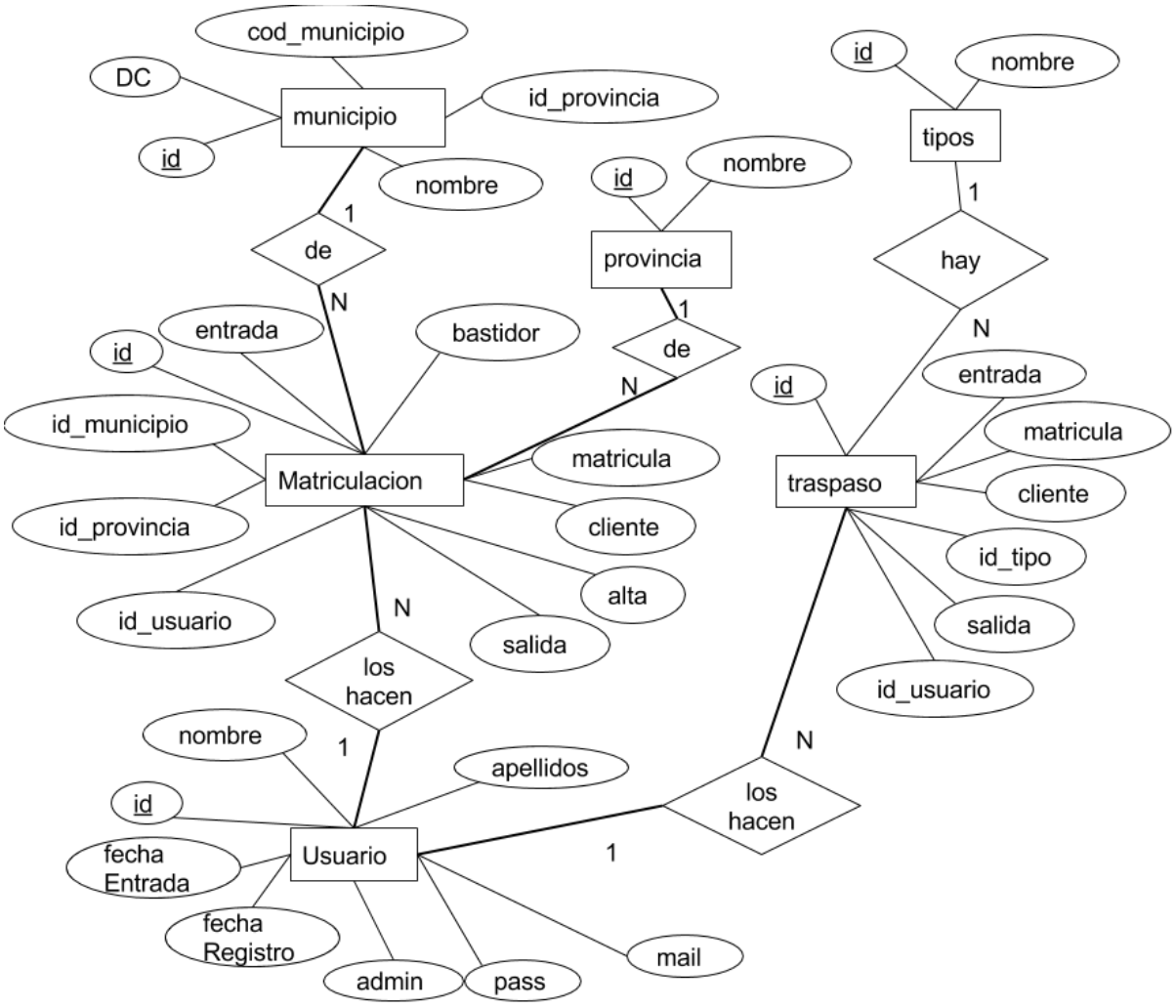


4.3 - Analisis

4.3.1 - Modelo Relacional



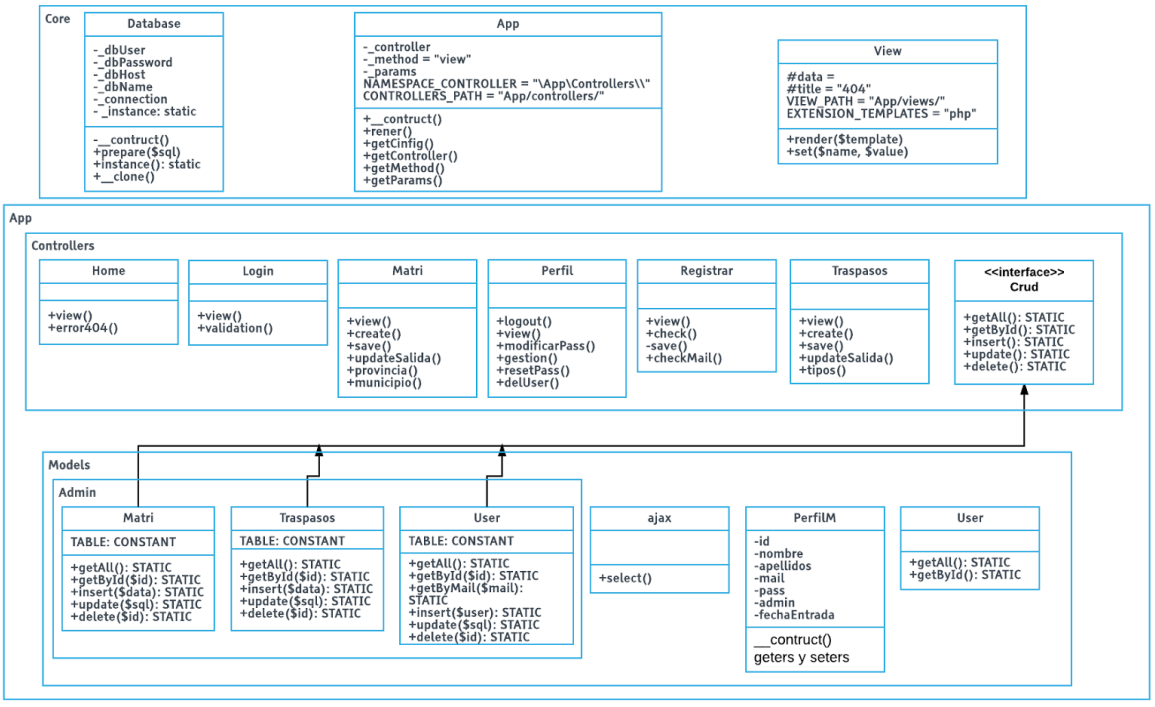
4.3.2 - Diagrama Entidad Relación



4.4 - Diseño

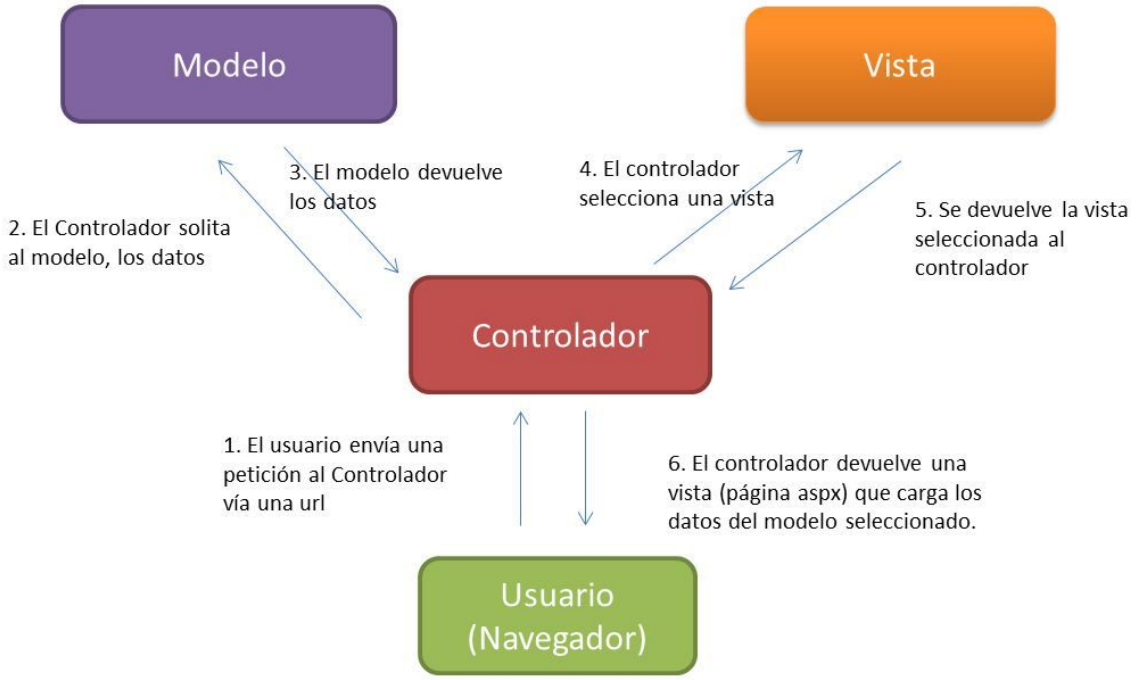
4.4.1 - Diagrama de clases

Como se puede observar, no existen herencias entre clases. Esto es debido al uso de namespace y use, que permiten crear visibilidad entre clases.
Como se puede observar en el diagrama, he representado los namespace de cada tabla con contenedores, un ejemplo para activar la visibilidad del modelo Traspasos dentro del controlador Traspasos, seria usando el comando use y declarando la ruta de acceso, que en este caso sería use \App\Models\Admin\Traspasos, y de esta manera obtendremos acceso directo a la clase Traspasos de los modelos, pero nos encontraríamos con un conflicto al tener la clase traspasos con el mismo nombre que la que acabamos de incorporar, pero para esto tenemos una solución, que es el uso de alias tal y como vemos a continuación, Ejemplo: use \App\Models\Admin\Traspasos as TraspasosAdmin. De esta manera ahora accederemos a la clase Traspasos a través del nombre TraspasosAdmin evitando conflictos con la actual clase llamada Traspasos



4.4.2 Patrones

Esto es la idea principal de cualquier MVC, el usuario realiza una petición al controlador de manera directa o indirecta al navegar sobre la página y el controlador envía una vista, la cual puede haber recibido datos del modelo o no.
Un ejemplo puede ser la página de inicio, mayormente suelo ser un contenido fijo y no depende de la base de datos.
También existe el caso de solo acceder al modelo a través del controlador para realizar una inserción en la base de datos, aun que es aconsejable dejar un feedback en la vista que estamos para confirmar la realización de la acción.



4.5 - Implementación

Investigación del MVC. Desde que empecé en las prácticas, comencé a ver el funcionamiento de esta organización de directorios y me familiarice con uno en concreto, que es el que he intentado implementar en mi aplicación, aunque no haya logrado implementar todo lo que habría deseado por falta de conocimiento sobre la manera de intercambiar datos entre modelo vista y controlador, al final he obtenido el conocimiento suficiente para recoger los datos de los modelos y enviarselos a las vistas.

Dada esta introducción al principio del desarrollo de la aplicación, comienzo con los aspectos que han sido claves para la creación del MVC a mi gusto.

La base del MVC que cogí, usaba la url sin usar parámetro para la carga de los controladores y intente adecuarlo a esa manera de funcionar, pero me encontré con errores, supongo que al no usarlo de manera correcta y decidí cambiarlo por el acceso a los controladores a través de parámetros (?controller=nombreControlador&action=nombreMetodo) el cual me parece mucho más intuitivo y fácil de usar.

Esta base consta de 3 archivos cores:
1- Encargado de detectar qué controlador se desea cargar, declararlo y cargar la configuración de la base de datos, llamado App.php. Este archivo ha sido uno de los que ha recibido más modificaciones de los archivos cores. En un principio disponía del código necesario para cargar los controladores a través de la url realizando una segmentación de la url a partir de la carpeta que almacena el index principal, se suprimió todo ese código y se genero uno alternativo para recoger los datos a través de los parámetros de la url. Se añadió también la llamada al método error404 del controlador home al no encontrar el controlador o el método a cargar. Por último dejar claro que gracias a este archivo es posible el uso de namespace para la

declaración de cada clase con solo su nombre y poder repetir el nombre de una clase en distintas carpetas sin entrar en conflicto entre sí.

2- Encargado de la comunicación con la base de datos, tiene la peculiaridad de usar un patrón singleton para evitar la reiterada creación de este objeto, y ahorrar recursos de nuestro servidor. Su funcionalidad es sencilla, bloquea el uso del constructor desde fuera de la clase y se almacena la clase en una de sus propiedades a traves de un metodo publico, el cual declara sólo la primera vez el objeto, y el resto de veces recupera el objeto ya almacenado. Para la comunicación utiliza la extensión PDO de php, que desconocía totalmente su existencia, pero viene siendo el clásico archivo DataSource.php que usábamos junto al método msqli, y me pareció muy útil su utilización. Más adelante con los modelos daré más detalles sobre su utilización. Por último, detallar que utiliza el método de la clase App para obtener la configuración de la base de datos y así obtener acceso. Esta clase llamada Databases no ha sido modificada.

3- Por ultimo, esta la clase encargada de cargar las vistas llamada View, dispone de dos métodos, uno que carga la vista y otro que permite declarar variables sobre la vista cargada. En un principio cargaba la vista seleccionada y listo, pero yo quería implementar un layout el cual estuviera siempre presente se cargará la vista que se cargarse. Tengo que admitir que me costo dejarlo tal y como está ahora, aunque estoy seguro que habrá mejores maneras de hacerlo así se me ocurrio ami por mi cuenta.
En resumen use un método ob_start() que recoge los datos imprimidos y los almacena en una variable a traves del metodo ob_get_content(), de esa manera hice un include del menú y lo almacene en la variable $menu, lo mismo con la plantilla que se deseaba cargar, la guarde en la variable $content y al final incluí el layout el cual contemplaba recibir estas dos variables junto a una tercera para el título. Podría explicar más cosas pero creo que sería enrollarse demasiado, si observais el código se aprecia el resto.

Clases controladores:

Estas clases van a disponer de métodos accesibles desde cualquier sitio de la aplicación gracias a la clase App, estos métodos van a ser los encargados de recoger los datos de los modelos y enviarlos a las vistas, también se van a poder crear métodos para validar datos sin necesidad de comunicarse directamente con el modelo, pudiendo ser llamados a través de otro método.

El método que más se ve en estas clases es el de view(), que es el encargado de visualizar la sección principal de la clase.

Clases Modelos

Existen 3 clases principales que implementan la interface Crud, El cual obliga a disponer de los métodos necesarios para la creación, lectura, actualización y eliminación de contenido de la base de datos. Estos métodos están obligados a usar la clase Core\Database para la comunicación con la base de datos, la cual instancia un objeto PDO que incorpora muchas e

4.6 - Pruebas

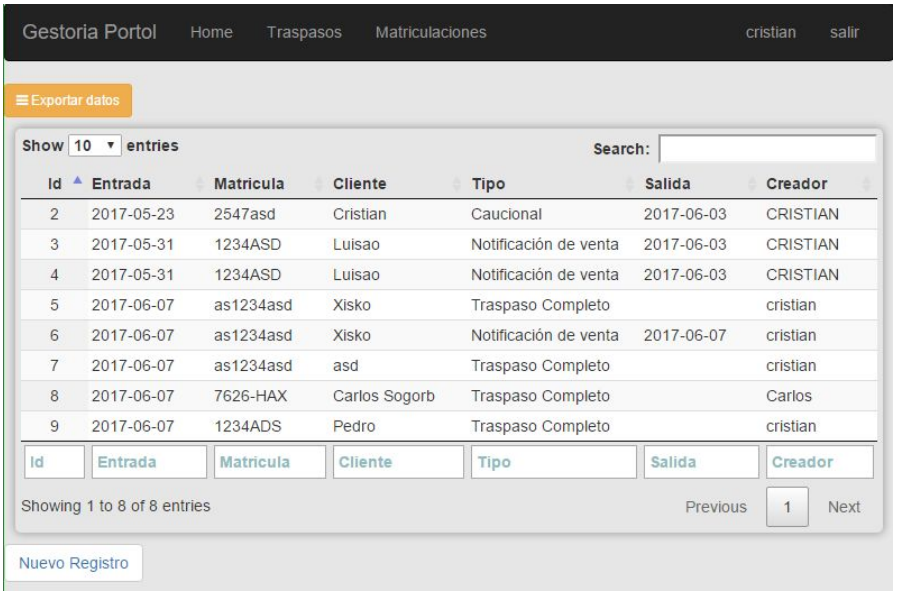
Ensayo y error

Al instalar la aplicación de dynamic DNS se realizó la prueba de visibilidad del dominio y no hubo respuesta del servidor, asique se investigó y se llegó a la conclusión que el router no dejaba pasar, se redirecciono el puerto 80 a mi ip local y se pudo comprobar que llegaba la petición a mi servidor apache, pero el apache bloqueaba el acceso, se procedió a dar acceso total a la carpeta del proyecto a través de httpd.conf cambiando la directiva require a “all granted” en ve de “local” y al final se consiguió tener acceso desde el exterior al servidor web. Tengo que decir que después de unos dias decidi actualizar windows para aumentar la seguridad del servidor, y al cabo de unos días sin verificar mi pagina desde el exterior, pude comprobar que no había acceso, realizando una prueba de ping desde un servidor externo, vi que mi router no respondía a tales peticiones,

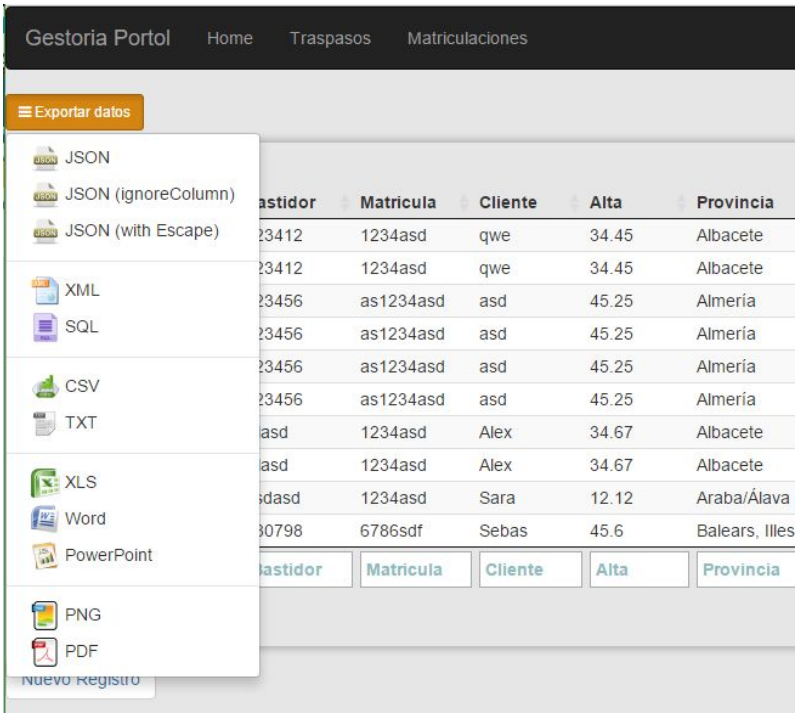
5 - Resultado y conclusiones

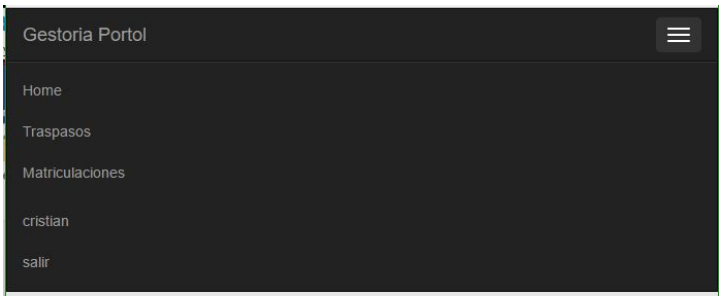
Resultados

Aquí se puede destacar el aspecto que crea la api de DataTables, realizando inputs por cada columna para su filtrado, y un filtrado general arriba a la derecha. A la vez cada columna se puede ordenar de manera ascendente o descendente.



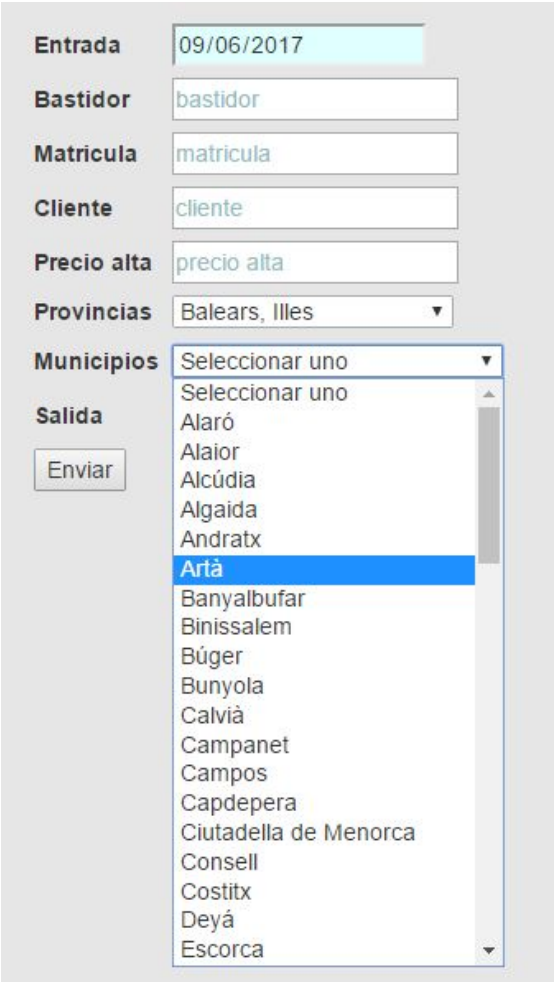
En esta screenshot se puede apreciar la implementación de la api exportData La cual permite exportar el contenido de la tabla en todos los formatos que se ven



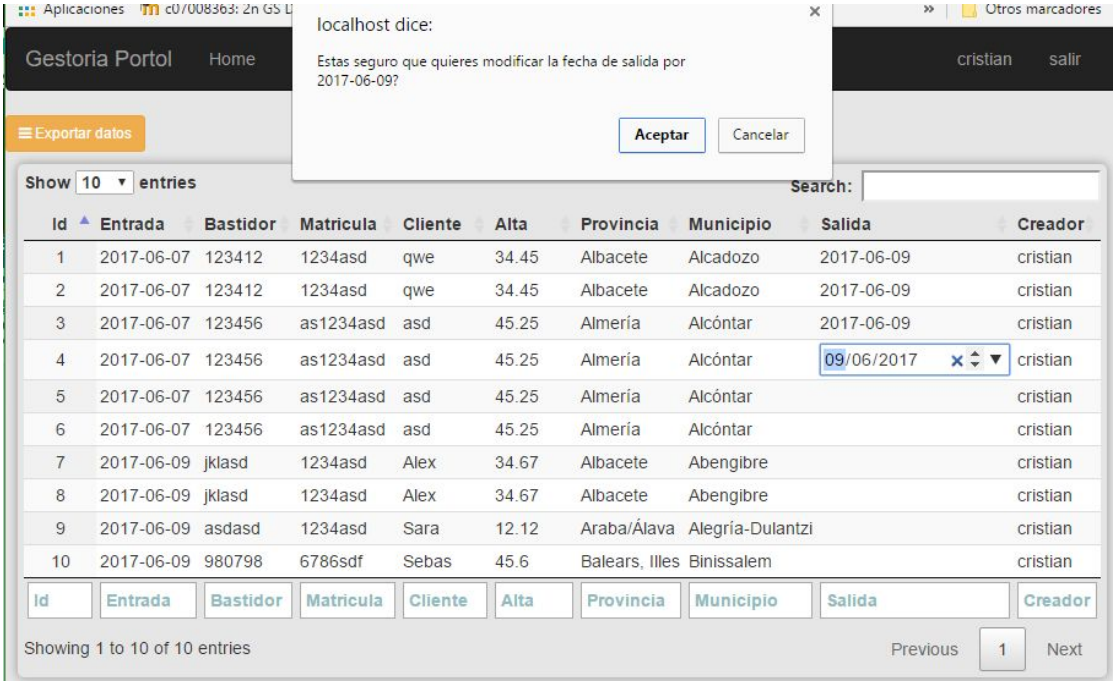


Menu responsive de bootstrap

Ejemplo de cómo se usa Ajax para rellenar el select de municipios, dependiendo de la opción seleccionada en el select de provincias



Uno de los aspectos que más me ha gustado es la posibilidad de editar la fecha de salida a través de la tabla como se realiza en phpmyadmin, se realiza doble click sobre el campo fecha salida que se desea modificar y se crea un input del tipo date, el cual al perder el foco, mostrará una confirmación y enviará un post via ajax de JQuery hacia un controlador, el cual se encargará de realizar la actualización en la base de datos y mediante JQuery se realizará la modificación en la tabla para evitar tener que recargar la página de nuevo y perder el filtrado de la tabla.



Conclusiones

Las conclusiones finales son que como siempre me he dedicado más a la funcionalidad de la aplicación, en este caso más en el patrón MVC, el cual me ha permitido sentirme más o menos cómodo en este nuevo patrón sabiendo prácticamente todo el proceso de comunicación entre el front y el back. El cual me permitirá crecer la aplicación sin ser muy engorroso el mantenimiento del código.

Por otra parte la aplicación se va a empezar a implementar desde el día 12 en la gestoría y se comenzará a poner a prueba, y en el momento que quede a gusto del cliente se venderá. Aun no tengo claro que podría pedir por ella ni cómo gestionar su mantenimiento, ya que podría pedir una cantidad única por la aplicación y el mantenimiento realizarlo a parte cobrando por él, o incluir el mantenimiento de cierto tiempo aumentando la cantidad.

También dejar claro, que en un principio se va ha instalar en una intranet, la cual no tendrá visibilidad desde el exterior, solo se podrá trabajar de forma local.

6 - Líneas de trabajo futuro

La aplicación está en su estado más básico, dispone de lo esencial para poder empezar a trabajar, y el MVC está listo para ir incorporando nuevas funcionalidades.

El siguiente paso sería añadir la posibilidad de agregar clientes desde un formulario extra y poder rescatarlos a la hora de introducir un nuevo registro en traspasos o matriculaciones, esta implementación añadiría una nueva tabla con su clave extranjera sustituyendo el actual campo cliente de cada tabla(Traspasos/Matriculaciones).

Después, se debería de implementar un sistema para enviar mensajes de texto y correos a cada cliente cuando al traspaso se le incorporar la fecha de salida, la cual indicará la disponibilidad de recogida de la nueva documentación.
Como siguiente punto, estaría bien empezar a incorporar una agenda para ir acumulando todos los clientes de la gestoría, y en caso de tener que notificar algún cambio de oficina, poder alertar a todos sus clientes con un simple clic.

Existe otra funcionalidad a implementar que la veo complicada pero no imposible.
Trata sobre la lectura de un determinado directorio para procesar la información de documentos word y redactar distintos documentos que incorporan datos segun la informacion leida de los word. Estos documentos redactados son agrupaciones de multas que hay me enviar a un mismo sitio. El objetivo final de esta funcionalidad es automatizar el proceso de rellenar manualmente listas con multas ha enviar a un mismo sitio.

7 - Bibliografia y webgrafia

Extract() http://php.net/manual/es/function.extract.php

namespace y use http://php.net/manual/es/language.namespaces.php

call_user_func_array() http://php.net/manual/es/function.call-user-func-array.php

PDO http://php.net/manual/es/class.pdo.php

parse_ini_file() http://php.net/manual/es/function.parse-ini-file.php

ob_Start() http://php.net/manual/es/function.ob-start.php

ob_get_contents() http://php.net/manual/es/function.ob-get-contents.php

ob_end_clean() http://php.net/manual/es/function.ob-end-clean.php

func_get_arg() http://php.net/manual/es/function.func-get-arg.php

setFetchMode() http://php.net/manual/es/pdostatement.setfetchmode.php

Fuente MVC basico:

https://www.uno-de-piera.com/desarrolla-tu-propio-mvc-con-php-y-poo/

Autor: Israel965

Api DataTables: https://datatables.net/ Autor: SpryMedia Ltd

Exportar tabla: http://ngiriraj.com/pages/htmltable_export/demo.php Autor: Giriraj

Bootstrap https://www.bootstrapcdn.com/ Desarroladores: getbootstrap.com

JQuery https://developers.google.com/speed/libraries/#jquery Autor: John Resig

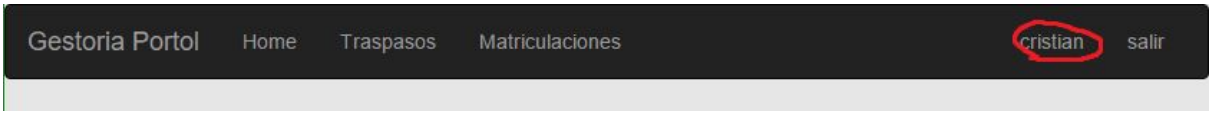
JQUERY UI https://developers.google.com/speed/libraries/#jquery-ui

Desarroladores: http://jqueryui.com/

8 - Anexo

Manual de usuario

Un usuario registrado dispone de 3 apartados para navegar, el primero es la posibilidad de modificar la contraseña a través de su nombre en el menú como se puede observar en el ejemplo



Después se abrirá una nueva página con los datos del perfil y con un botón para desplegar el cambio de contraseña

Una vez rellenado los campos correctamente, se mostrará un mensaje en color verde, de lo contrario, en caso de mostrarse en color rojo, significa que algo no se ha introducido de manera correcta como se muestra a continuación

A la hora de registrarse, hay varias validaciones, entre ellas existe un post via ajax de JQuery que se realiza siempre que se pierde el foco del campo mail, y en caso de existir el mail devuelve un mensaje indicando que ya existe el mail y no dejando abandonar el campo hasta encontrar un mail que no exista

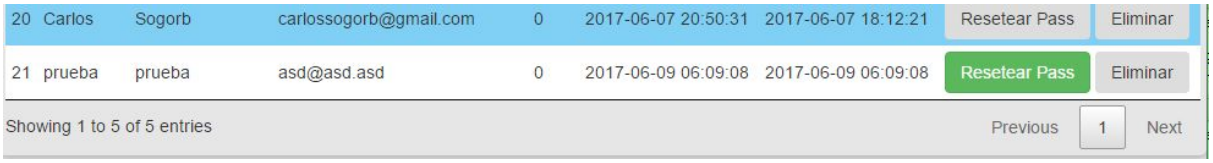
Para la inserción de un nuevo registro se debe entrar en la sección que se desee y clicar sobre el botón de nuevo registro.

Se abrirá una nueva vista con el formulario y si se inserta correctamente, nos redirige a la tabla para ver el resultado, de lo contrario, mostrará un error en el propio formulario

A medida que se vayan introduciendo los campos, si se validan el fondo del campo se pondrá de color azul

Manual para administrador

En la vista de gestión de usuarios, el botón resetear pass, deja como contraseña por defecto “Portol1”, En caso de aparecer el botón en verde, si aparece el botón en rojo, significa que ha habido algún problema



Documentación Técnica

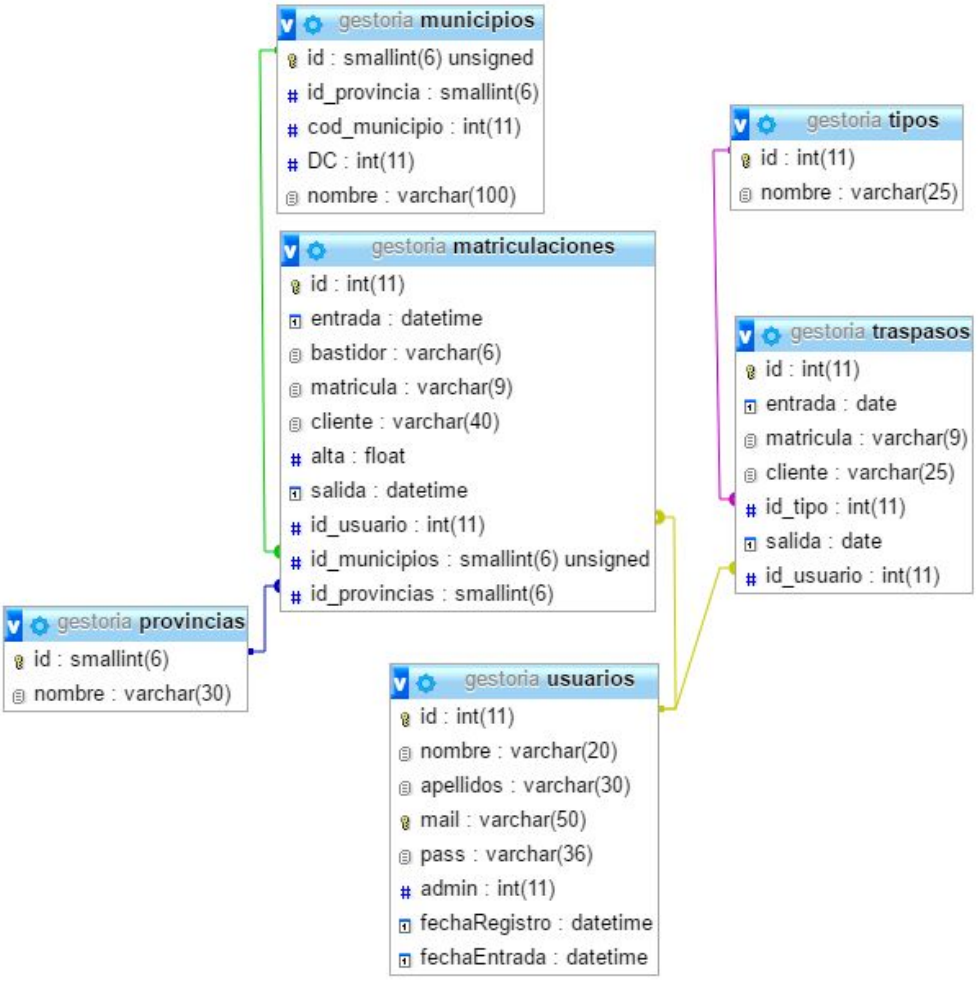
Un programador que venga a incorporar o mantener funcionalidades deberá saber que los controladores se cargan a través del parámetro controller poniendo como valor, el nombre de la clase y sus métodos se acceden con el parámetro action y poniendo como valor el nombre del método.
Para crear una nueva clase hay que incorporar su namespace para que el cargador de clases sea capaz de reconocerlo, si se implementa un controlador, su namespace sería App\Controllers, en caso de un modelo, sería App\Models, para un archivo core, Core. Este namespace coincide con su dirección relativa a partir de la raíz del proyecto.

En cada controlador se realiza una comprobación de una constante que se declara en el cargador de clases, y en caso de no encontrarla se deniega el acceso, de esta manera se evita el acceso directo a este controlador. Para los accesos restringidos de los invitados y registrados se realiza algo parecido, se comprueba en el archivo index.php de la raíz si existe una sesion iniciada, dependiendo de esto, se declara una constante INVITADO o USUARIO, la cual se comprobará su existencia en distintos archivos para dar o denegar acceso a algunas funcionalidades.

En los modelos, para acceder a la base de datos, hay que hacer uso de use \Core\Database para la comunicación con la base de datos. Hay que almacenar en una variable la instancia Database::instance(), quien contendrá el objeto PDO, el cual estará ya configurado con la base de datos, gracias al archivo config.ini

Por último destacar que cada controlador debe de usar la clase Core\View para poder acceder a las vistas y a la vez poder enviar datos, el método static set() permite enviar una variable con un valor, y el método render() muestra la vista.

Modelo Relacional sacado de phpmyadmin



La oración del día

PROGRAMA NUESTRO, QUE ESTÁS EN LOS SERVIDORES,
SANTIFICADO SEA TU CÓDIGO FUENTE; VENGA A NOSOTROS TU DOCUMENTACIÓN,
HÁGASE TU LÓGICA DE NEGOCIO, TANTO EN DESARROLLO COMO EN PRODUCCIÓN.

DANOS HOY NUESTROS REPORTES DE CADA DÍA;
Y PERDONA NUESTRAS MALAS PRÁCTICAS, ASÍ COMO NOSOTROS PERDONAMOS A LOS USUARIOS FINALES;
NO NOS DEJES CAER EN LA FRUSTRACIÓN,
Y LÍBRANOS DE TODO BUG.

#AMÉN